Master Universitario en Ingeniería Informática

Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

TESIS FIN de MASTER

DESARROLLO DE NUEVA FUNCIONALIDAD E INTERFAZ WEB Y MÓVIL PARA LA API MAPPINGAPI2

Autor:
Pablo Mosquera Díaz
Matricula:
U12A001
Tutor:
Miguel García Remesal

MADRID, 03 JULIO DE 2014

Dedicatoria:

Le dedico este trabajo a mis padres, por enseñarme siempre el buen camino en la vida y motivarme a lograr mis metas.

Agradecimientos:

Quisiera en primer lugar agradecerle al Departamento de informática Biomédica de la UPM por darme la oportunidad de ayudar en este gran proyecto. En especial a Miguel Ángel García y Alberto Anguita, gracias a vuestra incondicional ayuda he logrado culminar este trabajo.

Me gustaría agradecer a mi esposa, por ser el soporte en todo momento y motivación para lograr nuestras metas sin importar que tan difícil se pueda ver el camino en el horizonte.

También quisiera agradecer a toda mi familia, padres, suegros, abuelos, tíos y primos por brindarme todo el apoyo y ánimo en todos los momentos de mi vida y en especial durante la realización del Master. Y demostrarme que la familia siempre estará allí.

A mis amigos y compañeros de master por brindarme su ayuda en todo momento sin necesidad de pedirla.

Por ultimo darles las gracias a las Familias Feijo Spena y Díaz Mantopoulos por hacernos sentir, a mi esposa y a mí, integrantes más de vuestras familias y por brindarnos ayuda incondicional desde que llegamos a Madrid.

Resumen:

Vivimos en la era de la información y del internet, tenemos la necesidad cada vez mayor de conseguir y compartir la información que existe. Esta necesidad se da en todos los ámbitos existentes pero con más ahínco probablemente sea en el área de la medicina, razón por la cual se llevan a cabo muchas investigaciones de distinta índole, lo cual ha llevado a generar un cantidad inimaginable de información y esta su vez muy heterogénea, haciendo cada vez más difícil unificarla y sacar conocimiento o valor agregado.

Por lo cual se han llevado a cabo distintas investigaciones para dar solución a este problema, quizás la más importante y con más crecimiento es la búsqueda a partir de modelos de ontologías mediante el uso de sistemas que puedan consultarla.

Este trabajo de Fin de Master hace hincapié es la generación de las consultas para poder acceder a la información que se encuentra de manera distribuida en distintos sitios y de manera heterogénea, mediante el uso de una API que genera el código SPARQL necesario. La API que se uso fue creada por el grupo de informática biomédica.

También se buscó una manera eficiente de publicar esta API para su futuro uso en el proyecto p-medicine, por lo cual se creó un servicio RESTful para permitir generar las consultas deseadas desde cualquier plataforma, haciendo en esto caso más accesible y universal.

Se le dio también una interfaz WEB a la API que permitiera hacer uso de la misma de una manera más amigable para el usuario.

Abstract:

We live in the age of information and Internet so we have the need to consult and share the info that exists. This need comes is in every scope of our lives, probably one of the more important is the medicine, because it is the knowledge area that treats diseases and it tries to extents the live of the human beings. For that reason there have been many different researches generating huge amounts of heterogeneous and distributed information around the globe and making the data more difficult to consult.

Consequently there have been many researches to look for an answer about to solve the problem of searching heterogeneous and distributed data, perhaps the more important if the one that use ontological models.

This work is about the generation of the query statement based on the mapping API created by the biomedical informatics group.

At the same time the project looks for the best way to publish and make available the API for its use in the p-medicine project, for that reason a RESTful API was made to allow the generation of consults from within the platform, becoming much more accessible and universal available.

A Web interface was also made to the API, to let access to the final user in a friendly way.

CONTENIDO

1 Introducción

1.1 Motivación del proyecto:

Hoy en día la medicina y la informática se podría decir que trabajan juntas, existen diversas empresas y centros de investigación de informática biomédica. Quizás entre las empresas más conocidas a nivel mundial no exactamente por ser una empresa de informática biomédica, es Apple inc. Apple en su última versión del sistema operativo móvil IOS8, va a contar con aplicaciones nativas para la gestión de la salud, esto demuestra que empresas de informática están apostando cada vez más por la gestión de la salud.

Que las grandes empresas apuesten por la salud, da a entender la necesidad que existe por seguir adelantando en este ámbito, que aún le queda mucho por desarrollar y que traerá grandes beneficios a la sociedad.

Existen cada vez más pacientes en todo el mundo, los cuales necesitan cuidados más especializados. Se podría decir que el tratamiento para muchos de estos pacientes sería mucho mejor si fuesen personalizados.

Los tratamientos a medida son costosos, para abaratar costos, se presentan proyectos de bioinformática, los cuales tienen como fin consultar información necesaria para realizar los tratamientos a medida, consultar toda la información médica, es una tarea difícil debido a la heterogeneidad de los datos, y las prácticamente infinitas fuentes de consulta.

Dado lo difícil de las búsquedas, se han llevado a cabo proyectos que buscan integrar la información, proyectos como p-medicine.

1.2 Objetivos:

Ante la situación descrita en el apartado de motivación del proyecto, se plantea como trabajo de fin de master la propuesta de añadir opciones a la API de generación de consultas.

Por ello, los objetivos de este trabajo de fin de master son los siguientes:

- Estudiar a fondo la MappingAPI2, el cual es un paso necesario, ya que es la API que se utiliza para realizar consultas a partir de archivos mapeos los cuales contienen información sobre los metadatos y las clases que intervienen.
- Analizar la estructuración del lenguaje de consulta SPARQL, debido a que es el lenguaje de consulta que genera la API siendo estas las consultas que se usaran en el proyecto p-medicine.
- Conocer diferentes métodos que permitan ofrecer información para su posterior consumo, es necesario para que la información de la API pueda ser consumida por distintos tipos de clientes.
- Realizar un desarrollo que permita expandir la funcionalidad de la generación de las consultas de la mappingAPI2, este desarrollo permitirá generar consultas entre diferentes entradas de un mismo archivo de mapeo.
- Elaborar un prototipo funcional de una interfaz web, para darle acceso al usuario a la generación de las consultas SPARQL.
- Implantar la solución propuesta de mejora de la API, para el justo aprovechamiento de la API en un futuro por parte del proyecto p-medicine.

1.3 Descripción del trabajo

A continuación se explica brevemente el contenido de cada capítulo:

En el capítulo 1, Introducción, se explica la motivación y objetivos de este proyecto de fin de master.

En el capítulo 2, Estado del Arte, se exponen los aspectos técnicos relacionados con este proyecto de fin de master, al igual que las tecnologías más usadas en los distintos ambitos que abarca este trabajo.

En el capítulo 3, Evaluación de riesgos, se pretende demostrar las diferentes soluciones que se consideraron para llevar a cabo el proyecto de la manera más satisfactoria, se tomaron en cuenta ventajas y desventajas de cada posible solución.

En el capítulo 4, Resultados, En este capítulo se explica paso a paso las actividades que se realizaron para conseguir finalizar el proyecto junto con el resultado final.

En el capítulo 5, Conclusiones, En este capítulo se presentan las conclusiones del trabajo basándose en los resultados que se obtuvieron.

En el capítulo 6 Líneas Futuras, se presentan varias ideas como futuros objetivos que pueden surgir a partir de este trabajo de fin de master.

2 Estado del Arte.

En el presente capítulo, se explicara el ámbito que abarca el siguiente proyecto.

2.1 Proyecto p-medicine:

Proyecto que busca desarrollar nuevas herramientas de infraestructura de TI y modelos de VPH para acelerar la medicina personalizada para el beneficio del paciente.

La UPM es miembro del proyecto p-medicine, dentro del equipo de trabajo por parte de la UPM se encuentran, Prof. Dr. Víctor Maojo, Miguel García-Remesal y Alberto Anguita.

El rol principal de este equipo de trabajo es diseñar e implantar una interfaz de software para la integración semántica de datos biomédicos heterogéneos.

Objetivos del proyecto p-medicine:

- Crear un ambiente de colaboración que facilite el modelado de VPH.
- Combinar datos biológicos, clínicos, moleculares y genómicos en pacientes.
- Implementar los ensayos clínicos de adaptación y validación para el VPH dando lugar a apoyo en la toma de decisiones.
- Crear un DataWarehouse y un workbench capaz de ejecutar simulaciones de VPH.
- Sacarle provecho a la computación de alto rendimiento y al almacenamiento en la nube.
- Incrementar la calidad de la minería de datos en la investigación biomédica.
- Establecer un marco de servicios para acceder a recursos de materiales biológicos.
- Desarrollar un plan de negocio para hacer de p-medicine una entidad auto sustentada.

2.2 INTELIGENCIA ARTIFICIAL

La inteligencia artificial es un campo de la ciencia y la ingeniería que se ocupa de la comprensión desde el punto de vista informático, de lo que se denomina comúnmente comportamiento inteligente. También se ocupa de la creación de artefactos que exhiben este comportamiento.

Entre otras disciplinas de la inteligencia artificial tenemos:

- Procesamiento de lenguaje natural
- Visión artificial
- Resolución de problemas
- Representación del conocimiento y razonamiento
- Aprendizaje
- Robótica

2.4 Tecnología:

2.4.1 MappingApi2

Alberto, Ana y Andoni definen la mappingAPI2 en el manual de la mappingAPI2, como una API Java que ofrece una interfaz para editar y buscar anotaciones de las bases de datos del proyecto p-medicine. Contiene varias clases que permiten la construcción de objetos con la información en las anotaciones de la base de datos.

La MappingApi2, es la base de este proyecto de fin de Master, es una API que se ha venido desarrollando para un proyecto interuniversitario de investigación llamado p-medicine, cuyo fin es ayudar en la elaboración de tratamiento personalizados para pacientes. La Mapping hace de puente entre la información distribuida entre fuentes heterogéneas y el usuario que desea consultar dichas fuentes.

Esta API define un mapeo como la información que describe el alineamiento entre dos esquemas RDF uno físico (considerado la base de datos) y el otro conceptual (considerada la ontología que describe el dominio).

A continuación, mostramos un diagrama de clases de la API.

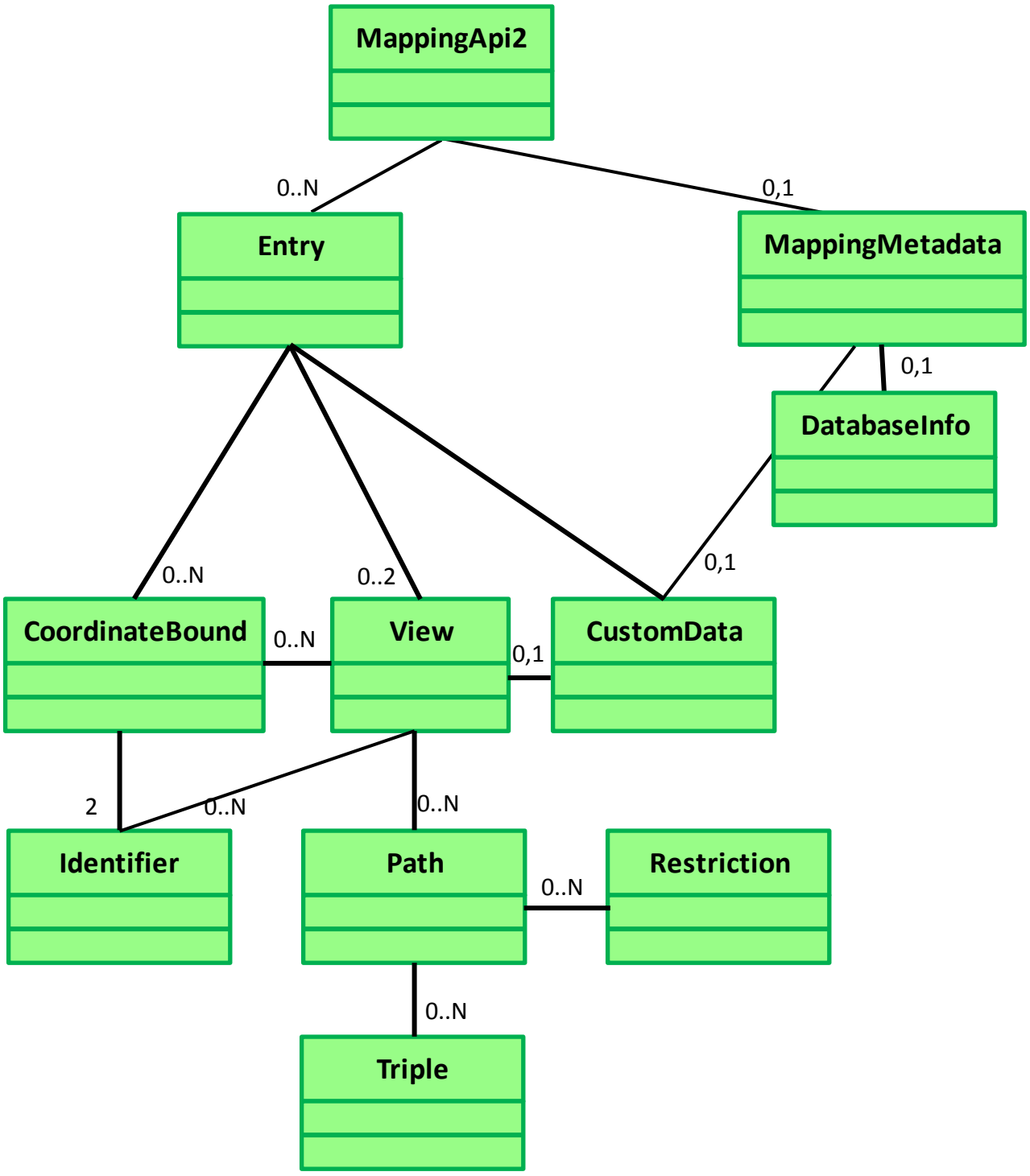
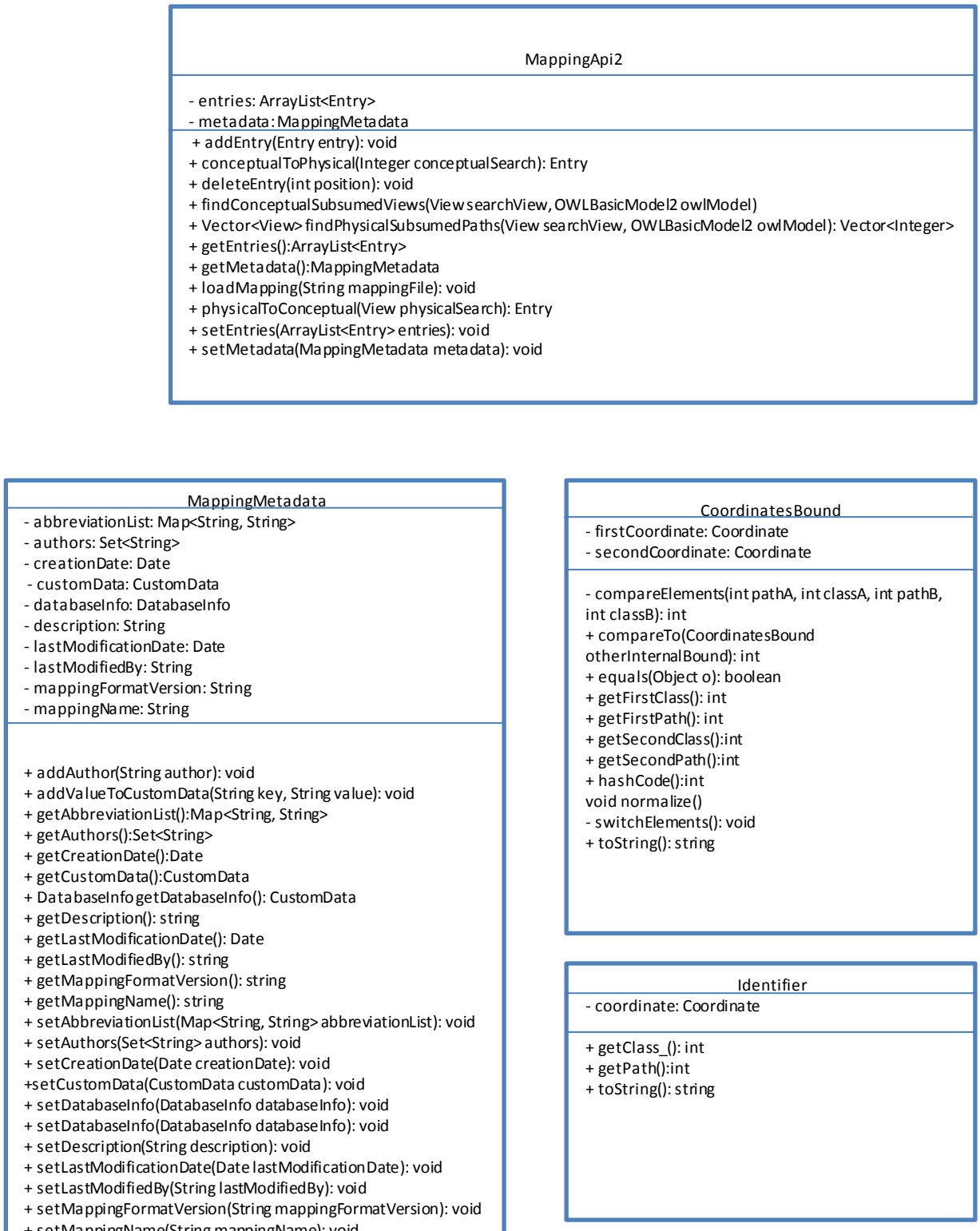


Diagrama de clases versión reducida de la mappingAPI2.

A continuación cada clase del diagrama en detalle.


MappingApi2
- entries: ArrayList<Entry>
- metadata: MappingMetadata
+ addEntry(Entry entry): void
+ conceptualToPhysical(Integer conceptualSearch): Entry
+ deleteEntry(int position): void
+ findConceptualSubsumedViews(View searchView, OWLBasicModel2 owlModel)
+ Vector<View> findPhysicalSubsumedPaths(View searchView, OWLBasicModel2 owlModel): Vector<Integer>
+ getEntries():ArrayList<Entry>
+ getMetadata():MappingMetadata
+ loadMapping(String mappingFile): void
+ physicalToConceptual(View physicalSearch): Entry
+ setEntries(ArrayList<Entry> entries): void
+ setMetadata(MappingMetadata metadata): void
MappingMetadata
- abbreviationList: Map<String, String>
- authors: Set<String>
- creationDate: Date
- customData: CustomData
- databaseInfo: DatabaseInfo
- description: String
- lastModificationDate: Date
- lastModifiedBy: String
- mappingFormatVersion: String
- mappingName: String
+ addAuthor(String author): void
+ addValueToCustomData(String key, String value): void
+ getAbbreviationList():Map<String, String>
+ getAuthors():Set<String>
+ getCreationDate():Date
+ getCustomData():CustomData
+ DatabaseInfo getDatabaseInfo(): CustomData
+ getDescription(): string
+ getLastModificationDate(): Date
+ getLastModifiedBy(): string
+ getMappingFormatVersion(): string
+ getMappingName(): string
+ setAbbreviationList(Map<String, String> abbreviationList): void
+ setAuthors(Set<String> authors): void
+ setCreationDate(Date creationDate): void
+setCustomData(CustomData customData): void
+ setDatabaseInfo(DatabaseInfo databaseInfo): void
+ setDatabaseInfo(DatabaseInfo databaseInfo): void
+ setDescription(String description): void
+ setLastModificationDate(Date lastModificationDate): void
+ setLastModifiedBy(String lastModifiedBy): void
+ setMappingFormatVersion(String mappingFormatVersion): void
+ setMappingName(String mappingName): void
CoordinatesBound
- firstCoordinate: Coordinate
- secondCoordinate: Coordinate
- compareElements(int pathA, int classA, int pathB, int classB): int
+ compareTo(CoordinatesBound otherInternalBound): int
+ equals(Object o): boolean
+ getFirstClass(): int
+ getFirstPath(): int
+ getSecondClass():int
+ getSecondPath():int
+ hashCode():int
void normalize()
- switchElements(): void
+ toString(): string
Identifier
- coordinate: Coordinate
+ getClass_(): int
+ getPath():int
+ toString(): string

+ Entry generateQueryIntegration(Entry entrada2)

- autor: String
- conceptualView: View
- creationDate: Date
- customData: CustomData
- description: String
- extBounds = new ArrayList<CoordinatesBound>
- lastModificationDate: Date
- lastModifiedBy: String
- physicalView: View

+ addExternalBound(CoordinatesBound bound): void
codifyPosition(int path, int class): string
+ compareMappingEntryConceptual(MappingApi2 mapping2, int indexEntry): boolean
+ compareMappingEntryPhysical(MappingApi2 mapping2, int indexEntry): boolean
+ equals(Object entry): boolean
+ Entry generateQueryIntegration(Entry entrada2)
+ generateQueryTriple(StringBuilder tripleBuilder, String Origin, String Relation, String Destination): void
- generateSPARQL(ArrayList<String> optionals, boolean conceptual): string
+ generateSPARQLFromConceptual(ArrayList<String> optionals): string
+ generateSPARQLFromConceptual():string
+ generateSPARQLFromPhysical():string
+ generateSPARQLFromPhysical(ArrayList<String> optionals)
- generateVariableName(int pathNum, int classNum): string
+ getAuthor(): string
+ getConceptualIdentifiers(): ArrayList<Identifier>
+ getConceptualView(): View
+ getCreationDate(): Date
+ getCustomData(): CustomData
+ getDescription(): string
+ getExternalBounds(): ArrayList<CoordinatesBound>
+ getLastModificationDate(): Date
+ getLastModifiedBy(): string
+ getPhysicalIdentifiers(): ArrayList<Identifier>
+ getPhysicalView(): View
+ getQueryExternalBounds(): Map<String, String>
+ isConceptualIdentifier(Identifier identifier): boolean
+ isPhysicalIdentifier(Identifier identifier): boolean
+ isSelectClass(String clase): boolean
+ setAuthor(String author): void
+ setConceptualView(View conceptualView): void
+ setCreationDate(Date creationDate): void
+ setCustomData(CustomData customData): void
+ setDescription(String desc): void
+ setExternalBounds(ArrayList<CoordinatesBound> extBounds): void
setFilters(Path path, int pathNum, StringBuilder filters): void
+ setLastModificationDate(Date lastModificationDate): void
+ void setLastModifiedBy(String lastModifiedBy)
+ setPhysicalView(View physicalView): void
+ setRestriction(ArrayList<CoordinatesBound> bounds): void
+ toString(): string

CustomData

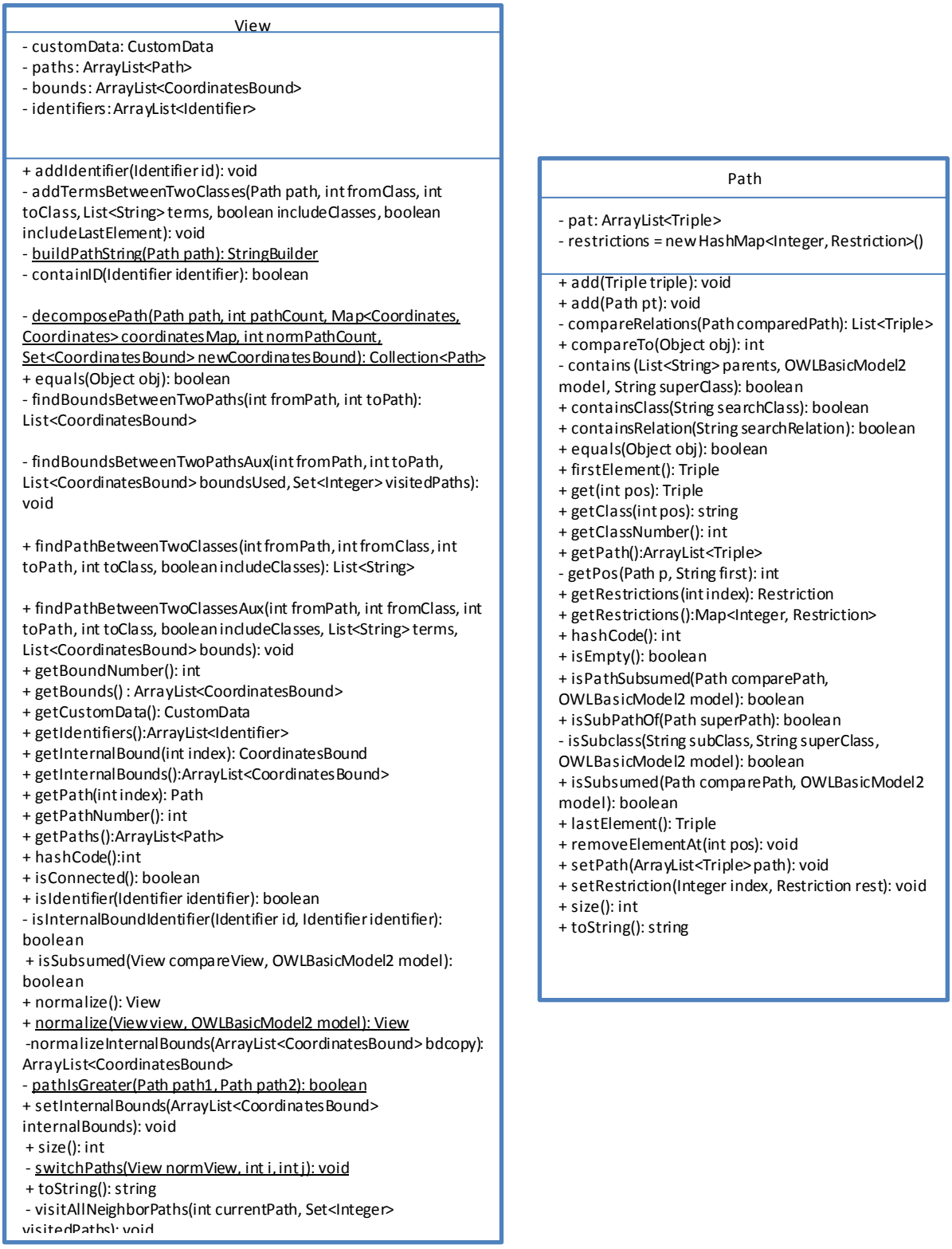
- values():Map<String, String>

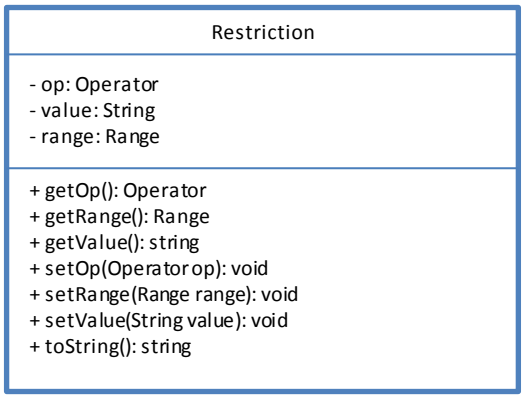
+ addValue(String key, String value): void
+ contains(String key): boolean
+ equals(Object o): boolean
+ get(String key): string
+ getData(): Map<String, String>
+ int hashCode()
+ remove(String key): void
+ setData(Map<String, String> data): void

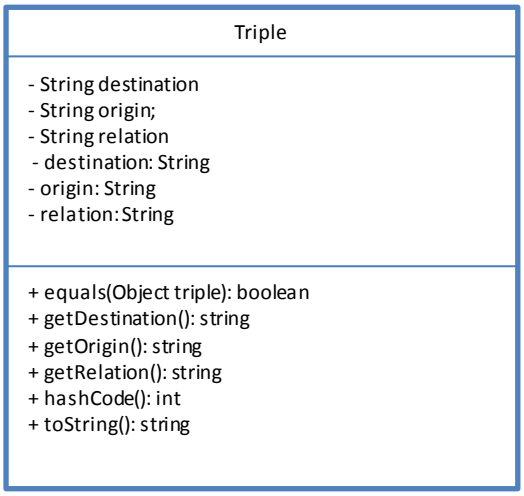
DatabaseInfo

- dbDescription: String
- dbId: String
- dbName: String
- dbPsw: String
- dbUrl: String
- dbUsr: String

+ getDbDescription(): string
+ getDbId():string
+ getDbName():string
+ getDbPsw():string
+ getDbUrl():string
+ getDbUsr():string
+ setDbDescription(String dbDescription): void
+ setDbId(String dbId): void
+ setDbName(String dbName): void
+ setDbPsw(String dbPsw): void
+ setDbUrl(String dbUrl): void
+ setDbUsr(String dbUsr): void


View
- customData: CustomData
- paths: ArrayList<Path>
- bounds: ArrayList<CoordinatesBound>
- identifiers: ArrayList<Identifier>
+ addIdentifier(Identifier id): void
- addTermsBetweenTwoClasses(Path path, int fromClass, int toClass, List<String> terms, boolean includeClasses, boolean includeLastElement): void
- buildPathString(Path path): StringBuilder
- containID(Identifier identifier): boolean
- decomposePath(Path path, int pathCount, Map<Coordinates, Coordinates> coordinatesMap, int normPathCount, Set<CoordinatesBound> newCoordinatesBound): Collection<Path>
+ equals(Object obj): boolean
- findBoundsBetweenTwoPaths(int fromPath, int toPath): List<CoordinatesBound>
- findBoundsBetweenTwoPathsAux(int fromPath, int toPath, List<CoordinatesBound> boundsUsed, Set<Integer> visitedPaths): void
+ findPathBetweenTwoClasses(int fromPath, int fromClass, int toPath, int toClass, boolean includeClasses): List<String>
+ findPathBetweenTwoClassesAux(int fromPath, int fromClass, int toPath, int toClass, boolean includeClasses, List<String> terms, List<CoordinatesBound> bounds): void
+ getBoundNumber(): int
+ getBounds() : ArrayList<CoordinatesBound>
+ getCustomData(): CustomData
+ getIdentifiers():ArrayList<Identifier>
+ getInternalBound(int index): CoordinatesBound
+ getInternalBounds():ArrayList<CoordinatesBound>
+ getPath(int index): Path
+ getPathNumber(): int
+ getPaths():ArrayList<Path>
+ hashCode():int
+ isConnected(): boolean
+ isIdentifier(Identifier identifier): boolean
- isInternalBoundIdentifier(Identifier id, Identifier identifier): boolean
+ isSubsumed(View compareView, OWLBasicModel2 model): boolean
+ normalize(): View
+ normalize(View view, OWLBasicModel2 model): View
-normalizeInternalBounds(ArrayList<CoordinatesBound> bdcopy): ArrayList<CoordinatesBound>
- pathIsGreater(Path path1, Path path2): boolean
+ setInternalBounds(ArrayList<CoordinatesBound> internalBounds): void
+ size(): int
- switchPaths(View normView, int i, int j): void
+ toString(): string
- visitAllNeighborPaths(int currentPath, Set<Integer> visitedPaths): void
Path
- pat: ArrayList<Triple>
- restrictions = new HashMap<Integer, Restriction>()
+ add(Triple triple): void
+ add(Path pt): void
- compareRelations(Path comparedPath): List<Triple>
+ compareTo(Object obj): int
- contains (List<String> parents, OWLBasicModel2 model, String superClass): boolean
+ containsClass(String searchClass): boolean
+ containsRelation(String searchRelation): boolean
+ equals(Object obj): boolean
+ firstElement(): Triple
+ get(int pos): Triple
+ getClass(int pos): string
+ getClassNumber(): int
+ getPath():ArrayList<Triple>
- getPos(Path p, String first): int
+ getRestrictions(int index): Restriction
+ getRestrictions():Map<Integer, Restriction>
+ hashCode(): int
+ isEmpty(): boolean
+ isPathSubsumed(Path comparePath, OWLBasicModel2 model): boolean
+ isSubPathOf(Path superPath): boolean
- isSubclass(String subClass, String superClass, OWLBasicModel2 model): boolean
+ isSubsumed(Path comparePath, OWLBasicModel2 model): boolean
+ lastElement(): Triple
+ removeElementAt(int pos): void
+ setPath(ArrayList<Triple> path): void
+ setRestriction(Integer index, Restriction rest): void
+ size(): int
+ toString(): string





Cada clase se explicara en mayor detalle a continuación.

- MappingApi2: Es la principal clase de la API, guarda la información acerca de un mapeo, los objetos de esta clase se pueden crear vacios o a partir deun archivo XML de mapeo.

- MappingMetadata: Contiene metadata de un mapeo, es información adicional, por ejemplo el nombre la descripción del lenguaje natural.

- DatabaseInfo: Contiene Meta información de la base de datos física.

- Entry: Un entrada está compuesta por 2 vistas, una physical y otra conceptual, y representa la equivalencia semántica entre dos vistas RDF.

- View: Representa una vista de un modelo RDF, estas vistas pueden ser physical o conceptual, las vistas están compuestas por Paths.

- Identifier: Esta clase sirve para etiquetar alguna de las clases RDF como clases de identificación.
- Path: Representa el camino formado por una concatenación de tripletas.

- Triple: Representa una tripleta RDF es decir está compuesto por un sujeto o clase, un predicado o relación y un objeto o tipo de dato.

- CoordinatesBound: Representa el vínculo entre dos clases RDF, estas pueden ser internas o externas, internas cuando el vínculo es entre dos clases de la misma

vista y externa cuando el vínculo es entre dos clases que se encuentran en diferentes vistas.

- Reestriction: Representa las restricciones que pueden tener las clases del tipo DataTypes, en cuanto al valor que deben que deben buscar, por ejemplo que un valor numérico sea mayor (>) que cinco (5).

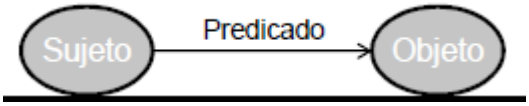
2.4.2 RDF

Según Juan Antonio Pastor en el libro Tecnologías de la web semántica define RDF como "un modelo para la descripción estructurada de recursos de información en internet. El modelo RDF trata de dar respuesta a ciertos problemas planteados por la aplicación de XML, dentro de la búsqueda de soluciones más flexibles y eficientes para la representación de información. Esto es, el modelo jerárquico de organización de datos XML resulta muy limitado.".

Se afirma también que la unidad de información que se utiliza para describir en RDF es la tripleta. Una tripleta está compuesta por:

- Sujeto.
- Predicado.
- Objeto.

Gráficamente sería algo como:



2.4.3 SPARQL

El consorcio W3C define SPARQL como un lenguaje para la consulta de grafos RDF, cuya misión es obtener tripletas del grafo RDF basándose en una comparación con una tripleta definida.

2.4.4 Git

Según el portal web [10] Git es un sistema de control de versiones, diseñado por linus torvalds, busca versatilidad y está enfocado en el desarrollo de software de grande grupos. Cuenta con una gran variedad de comandos: fetch, merge, pull, commit, push, status, checkout, etc..

2.4.5 IDE

Según [4], Un IDE, es un entorno de desarrollo integrado, es decir es un programa informático compuesto por un conjunto de herramientas de programación.

Según la web buenastareas.com "Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Delphi, Visual Basic, etc. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos detexto, como es el caso de Smalltalk u Objective-C.

Es posible que un mismo IDE pueda funcionar con varios lenguajes de programación. Este es el caso de Eclipse, al que mediante plugins se le puedeañadir soporte de lenguajes adicionales."

2.5. Tecnología WEB cliente:

2.5.1 PHP

Para el desarrollo de este proyecto fue utilizado en su mayoría el lenguaje de programación PHP.

PHP son las siglas de *Hypertext Pre-processor*. Este lenguaje es definido por el portal php.net (2014) como "un lenguaje de scripting, el cual puede estar contenido dentro de páginas HTML…El objetivo del lenguaje es permitir a Desarrolladores Web escribir paginas generadas dinámicamente con rapidez.".

PHP es un lenguaje que es gratuito, no se tiene que pagar licencias, es altamente compatible con servidores de bases de datos, especialmente con *MySQL*.

2.5.2 JavaScript

La Universidad de Málaga (2014) en su sitio web define JavaScript como "un lenguaje interpretado que permite incluir macros en páginas Web. Estas macros se ejecutan en el ordenador del visitante…y no en el servidor" cuando se habla de ordenador del visitante se hace referencia al cliente.

Entre las utilidades que proporciona JavaScript encontramos, la comprobación de datos que son introducidos por el usuario antes de enviarlos al servidor y no depender del servidor para la realización de cálculos sencillos.

JavaScript al ejecutarse en el cliente, ayuda a que se descongestione un poco el servidor que normalmente está sobrecargado.

2.5.3 HTML

Las siglas HTML significan Lenguaje de Marcas de Hipertexto, el portal de [11] lo describe como "Lenguaje que sirve para modelar texto y agregarle funciones especiales. Es la base para la creación de páginas web tradicionales". Entre las funciones especiales que tiene HTML es que se puede usar junto a lenguajes Script como JavaScript y PHP.

2.5.4 XML

Es un lenguaje que sirve para estructurar, guardar o transportar información, por si solo XML no hace nada, cuyo significado en inglés es eXtensible Markup Language. Según [12]

Se podría decir que HTML es un tipo de archivo XML ya que XML abarca mucho más, en XML se crean las etiquetas que uno necesite para trabajar. Siendo el caso contrario el HTML el cual cuenta con las etiquetas predefinidas.

2.5.5 CSS

CSS son las siglas para Cascading Style Sheets u Hojas de Estilo en Cascada en español. El sitio web [13] lo definen como "la tecnología desarrollada con el fin de separar la estructura de la presentación". Es decir que describe como se mostrara el documento web por pantalla, el documento web estará escrito en HTML o XML.

Las Hojas de Estilo en Cascada funcionan en base a reglas, si alguna de estas reglas se modifica, la presentación final de las páginas que se basaban en esa Hoja de Estilo cambiara también, dejando a un lado la necesidad de cambiar el estilo hoja por hoja.

2.6. Tecnología WEB Servidor:

2.6.1 JAVA

Java es un lenguaje de programación creado por la ya extinta compañía informática sun microsystem. La página web [14] lo definen como "

Java es un lenguaje de programación con el que podemos realizar cualquier tipo de programa. En la actualidad es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general… La independencia de plataforma es una de las razones por las que Java es interesante para Internet, ya que muchas personas deben tener acceso con ordenadores distintos".

Java también es conocido por ser orientado a objetos, que se ejecuta en una Máquina Virtual lo cual hace que pueda ser ejecutado en cualquier plataforma que cuente con una máquina virtual java.

2.6.2 J2EE

J2EE también conocido como Java empresarial es una plataforma Java, es una plataforma independiente basada para el desarrollo de aplicaciones web empresariales en línea cuanta con su set de servicios, APIs y protocolos, según [15].

2.6.3 WAR File

El nombre WAR proviene de Web Archive, según [16] en castellano archivo web y es un archivo que contiene todo lo necesario para desplegar un proyecto WEB en Java, contiene las clases compiladas, archivos de configuracions, JSP, librerías, HTML´s, javascripts y css´s que constituyen un proyecto WEB.

2.6.4 Servidor

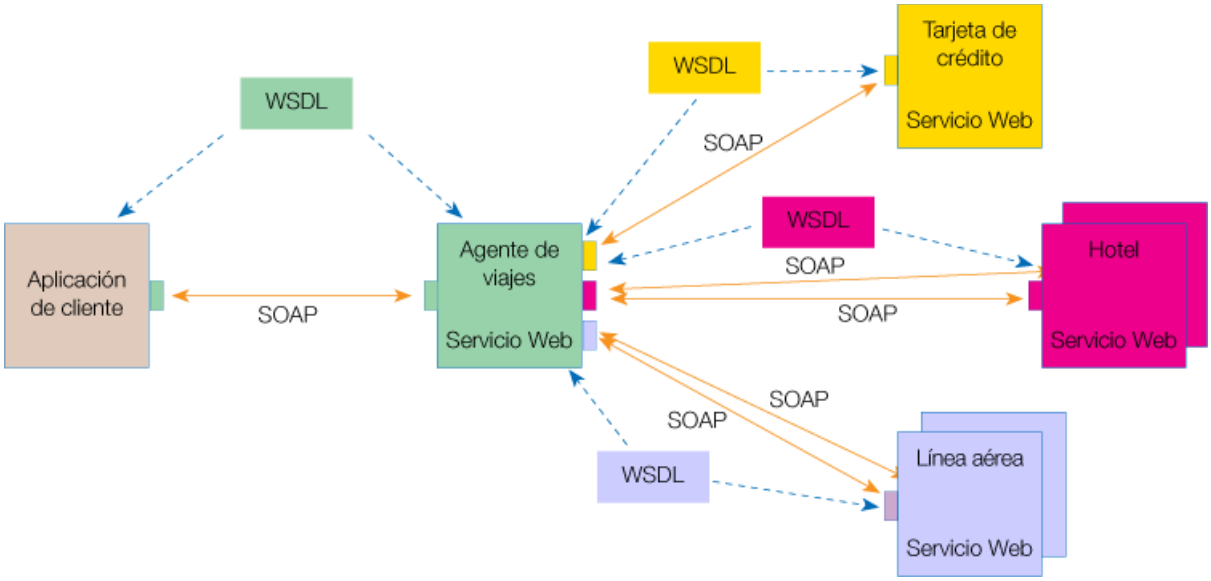
De acuerdo con Stallings (1999, p. 667) "En una red, una estación de datos que proporciona servicios a otras estaciones: por ejemplo un servidor de archivos, un servidor de impresión o un servidor de correo".

Quizas entre los servidores más comunes hoy en dia, estén los servidores web, existen infinidad de servidores, para todo tipo de plataformas, por ejemplo el servidor apache que funciona para PHP o servidores como GlassFish y JBoss que funcionan para Java.

Servicios WEB

Los servicios WEB es un concepto abstracto difícil de definir, según la W3C en su página web lo define como "un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer unos servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web.".

Este tipo de servicios son de gran ayuda ya que aportan valor agregado a los servicios que tienen, al poder ofrecerlos de una manera más eficiente y pudiendo usar varios de estos servicios, uniéndolos para dar nuevos servicios con valor agregado.

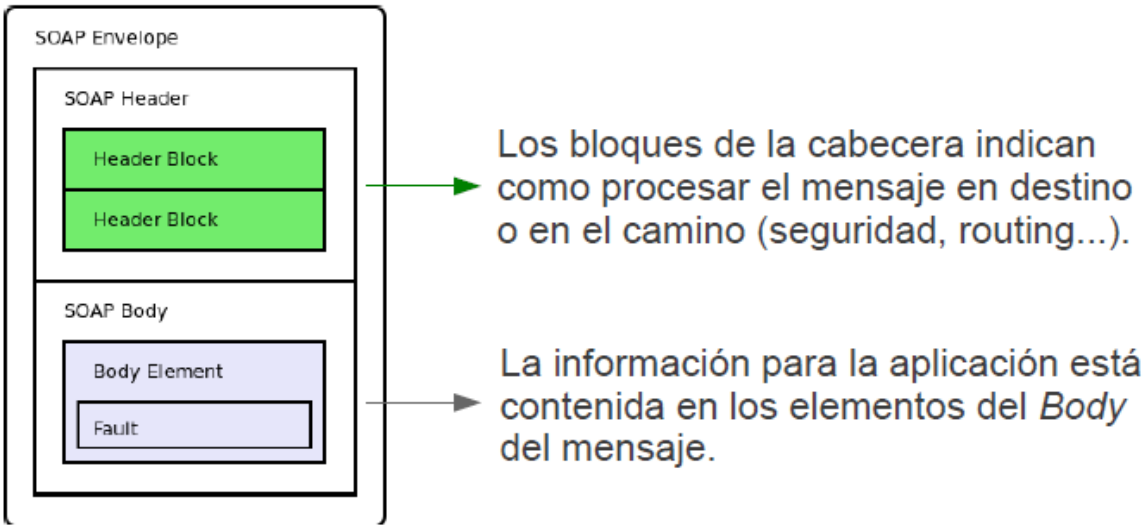


http://w3c.es/Divulgacion/GuiasBreves/ServiciosWeb

2.6.5 SOAP

SOAP es un protocolo ligero cuyo fin es el intercambio de información de una manera descentralizada, está basado en tecnología XML, este framework de mensajería fue

diseñado para ser independiente de la plataforma. Fue creado por un consorcio en el cual Microsoft formaba parte.

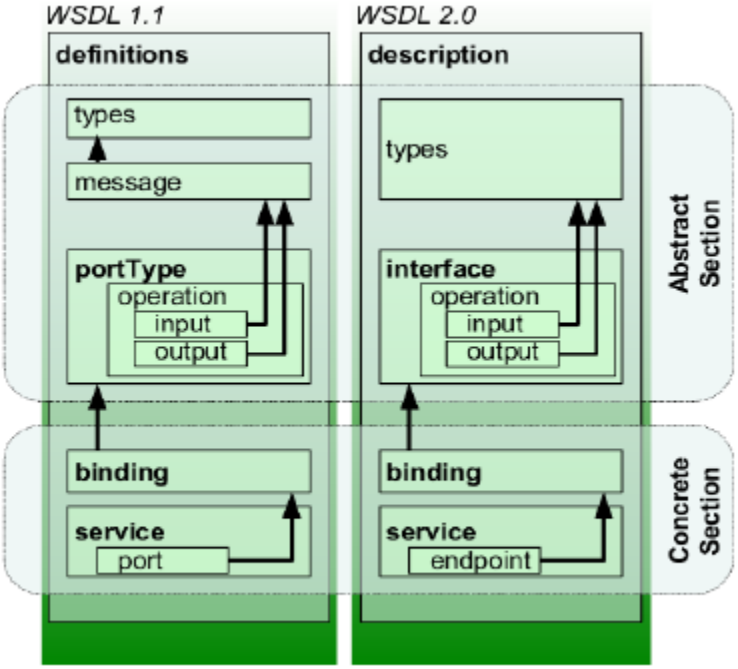


2.6.6 WSDL

Es el lenguaje de descripción de servicios WEB, esta expresado en la tecnología XML, en el de manera univoca se definen los tipos y métodos ofrecidos por el servicio WEB, los protocolos y el formato de los mensajes.

A partir del WSDL se puede generar el código necesario para acceder a los servicios, existen distintas librerías que lo generan automáticamente como Axis2 para java o zendFramework para PHP.

Partes del documento WSDL



2.6.7 Servicios RESTful

Se podría definir como una evolución de los servicios WEB, usado por las grandes compañías, La mayoría de los servicios que ofrecen estas empresas, es a través de un servicio RESTful. Tiene origen académico.

Según la web [17] es "Un estilo arquitectónico que, cuando se sigue, permite a los componentes realizar sus funciones maximizando las propiedades arquitectónicas más importantes de un sistema global de información basado en red (e.g. maximizar el incremento de la información identificada dentro del sistema, con lo que se incrementa la utilidad del sistema en su conjunto)"

Los servicios Rest se ofrecen por el protocolo HTTP, usando sus verbos correspondientes:

Método http	Es seguro	Es idempotente
GET	SI	SI
HEAD	SI	SI
OPTIONS	SI	SI
PUT	NO	SI
DELETE	NO	SI(*)
POST	NO	NO

Según Dr. Soriano en las transparencias de Sistemas orientados a servicios, "Un método se dice seguro cuando su ejecución no altera el estado del recurso y además no se espera que cause efectos colaterales no intencionados (operaciones desólo lectura)…. Un método se dice idempotente cuando su ejecución repetida sobre el mismo recurso provoca el mismo efecto que una única ejecución"

Para los Servicios Rest todo es un recurso.
También existen numerosas librerías que permiten una creación agil de servicios rest, por ejemplo entre las librerías REST para java más importantes están Jersey y RestEasy, según la web de jboss.org define resteasy como un proyecto que ayuda a contruir servicios RESTful para Aplicaciones Java.

2.6.8 CloudComputing

El cloud computing, en español computación en la nube hace referencia a los servicios ofrecidos a través de internet, estos servicios son variados, pueden ser SaaS, PaaS o IaaS. Dependiendo de cada particularidad.
El cloud computing está revolucionando la industria del internet ya que ofrece servicios con un único requerimiento que es tener conexión a internet, ya queel resto se encuentra almacenado en algún servidor de internet

2.6.9 Infraestructura como Servicio, IaaS

En ingles Infraestructure as a Service,es un servicio de hardware virtualizado en la nube, permite escoger a medida, el tipo de red, procesador, memoria entre otros, permitiendo

tener un hardware que se ajuste a las necesidades y únicamente pagando por consumo del mismo.

Un ejemplo de este tipo de servicio lo ofrece Amazon, con sus denominados Amazon Web Services, ofrecen almacenamiento e infraestructura.

2.6.10 Plataforma como Servicio, PaaS

Con sus siglas en Ingles PAAS , Platform as a Service, es un servicio que permite alojar aplicaciones, los datos y sus componentes, es un entorno con servicios añadidos como por ejemplo J2EE, también sirven como entornos de desarrollo.

Entre las principales plataformas que ofrecen servicios PaaS están:

- Google AppEngine.
- Windows Azure.
- Openshift de Red Hat
- Heroku.

2.6.11 Software como Servicio,SaaS

El software como servicio, es un programa que se ofrece por la nube, podria ser por ejemplo un suite office o un programa de correos, este tipo de servicios, es el más extendido de la nube, ya que ofrece mucha variedad de software y no necesita de un usuario técnico especializado que configure la plataforma o el hardware.

Ejemplos de software as a service, es por ejemplo office en la nube, el correo sin dercargarlo en local, CRMs en línea.

Existen infinidad de empresas que ofrecen estos servicios, podemos encontrar por ejemplo los suites de office de Google llamada GoogleDocs, o lasuite de office de Microsoft en la nube llamada Office 365.

Denoe.es

2.7 Tecnología Móvil:

2.7.1 PhoneGap:

Es un Framework que permite el desarrollo de aplicaciones móviles multiplataforma basándose en tecnologías Web HTML, CSS y JavaScript. Ya que el framework hace de puente entre las tecnologías web y tecnologia del dispositivo móvil. Permitiendo de esta manera a partir de un único tipo de tecnología generar diversas aplicaciones móviles en diferentes plataformas.

Beneficios de PhoneGap

- Se puede codificar en HTML5, CSS y JavaScript en vez de hacer uso del lenguaje de la plataforma como por ejemplo Objective C, Java o C#.
- La aplicación una vez lista puede hacerse uso de ella en varias plataformas a la vez.
- Permite hacer uso de las características internas del dispositivo móvil, como por ejemplo cámara o acelerómetro.
- Se pueden obtener Interfaces graficas mas limpias.
- Se puede distribuir la aplicación entre distintas tiendas móviles.
- Actualmente se PhoneGAp cuenta con 7 plataformas a las que ofrece soporte, iOS, Android, Blackberry, webOS, Symbian, Bada y WindowsPhone.
- Es Open Source.

Desventajas de PhoneGap

- Muchas veces no cuenta con el soporte para todas las especificaciones internas de los dispositivos móviles, por ejemplo USB port de algunos dispositivos.
- Siempre estara un paso más atrás del resto debido a que se debe desarrollar la funcionalidad que permite hacer de puente entre el dispositvo y el framework PhoneGap.
- Visibilidad un poco más genérica

3. **Evaluación de Riegos:**

Para demostrar que se han considerado diferentes soluciones tanto clásicas como novedosas o innovadoras al problema, y se han evaluado los riesgos y ventajas de cada una de ellas.

El primer riesgo dentro del proyecto fue decidir cómo implementar la mejora de las capacidades de la mappingAPI2, para desarrollar la mejora de la API, se tenían 2 propuestas:

1. Usar una nueva API de generación de SPARQL, llamada apisparql la cual es una librería especializada en SPARQL y entre sus métodos de interés está la generación de la consulta.
2. Realizar un nuevo método dentro de la API mappingAPI2, que permitiera unificar entradas y usar el método de generación con la que ya contaba la API.

Ambas propuestas conllevaban cierto riesgo la primera propuesta tenía el riesgo de tener que aprender a usar dos apis distintas e integrarlas, generando una gran inversión de tiempo para aprender a usarlas y unificar el código.
La segunda propuesta por otro lado, conllevaba una inversión de tiempo considerable también junto con la necesidad de programar nuevos métodos dentro de la mappingAPI2, los nuevos métodos no tenían tampoco una garantía de que funcionarían.

Ambas propuestas coincidían en la necesidad de aprender el funcionamiento total de la mappingAPI2, así que antes de decidirme por alguna de las propuestas, me decante por aprender esta API construida por el departamento de informatica biomedica.

Como segundo riesgo dentro del proyecto se tenía la necesidad de elaborar una Interfaz gráfica, preferiblemente WEB, para ofrecerle al usuario acceso a la API. Este segundo riesgo se barajearon en un principio dos propuestas.

1. Una página Web basada en tecnología Java que tuviera la mappingAPI2 integrada.
2. Una página Web basada en tecnología PHP que ejecutara el jar compilado pasándole los parámetros necesarios dependiendo del caso.

Ninguna de las propuestas anteriores me pareció lo sufrientemente buena para un proyecto de tan gran envergadura como lo es p-medicine, por lo cual después de analizar más a fondo se decidió hacer un servicio RESTful ofreciendo el servicio de generación de consulta y ya a partir de ese servicio se construyó una aplicación web que hace llamadas a este servicio y muestra los resultados.

A pesar de que no entraba dentro del scope del proyecto la construcción de un servicio RESTful, me pareció que encajaba muy bien en este tipo de proyecto, que seguirá creciendo y al tener una interfaz separada de la lógica se puede avanzar o mejorar una sin tener un impacto directo en el mantenimiento del proyecto a futuro.
Eliminando riesgos futuros y mejorando y ampliando el alcance de la API actual. Es decir a la larga se minimizaban los riesgos.

4 Resultados:

En este apartado se explicara detalladamente el proceso que se siguió para conseguir los objetivos de este trabajo de fin de master.

El primer paso que se dio, fue la reunión con el tutor que me comenta el proyecto, la idea del mismo es mejorar las capacidades de generación de Queries SPARQL de la mappingAPI2, que es una API que se desarrolló dentro del grupo de informática Biomédica de la antigua facultad de informática ahora escuela superior técnica superior de ingenieros informáticos de la universidad politécnica de Madrid. La API se ha venido desarrollando dentro de un proyecto en conjunto entre distintos centro de investigación europeos llamado p-medicine.

El estado de la mappingAPI2, era estable y funcionaba bien. Permitía generar Queries SPARQL de cualquier entrada de un archivo de mapeo.

La funcionalidad que se me pidió agregar fue la de permitir que se generaran queries SPARQL entre dos entradas distintas dentro del archivo de mapeo, para poder generar una quería que tomara en cuenta dos entradas, estas entradas deben tener clases en común que permitieran unirse.

Las clases para unir entradas distintas dentro de la mappingAPI2 tenían que cumplir tres condiciones:

1. Las dos clases debían ser el mismo tipo de clase.
2. Las dos clases debían contener external-Bounds.
3. Y no podían ser clases del tipo data-types. Es decir que no fueran clases de tipos de variables (String, Integer, Float…etc).

Partiendo de que es necesario que se cumplan los tres puntos anteriores, para que dos clases de distintas entradas puedan unirse, se empezó a analizar cuáles de las dos propuestas explicadas en el punto anterior de riesgos se podía llevar a cabo, ya fuese usar una nueva api llamada apisparql o crear un método que creara una entrada a partir de otras dos que pudiesen unirse y reutilizar el método de la mappingAPI2 de generar el código SPARQL de una entrada.

Al final, debido a que ambos enfoques conllevaban riesgos, y la información de las clases y métodos de la mappingAPI2 estaba muy bien descrita, preferí decantarme por la segunda propuesta de programar un método de unión de dos entradas en caso de que estas se pudiesen unificar.

Para lograr programar este método fue necesario aprender a fondo la funcionalidad de la mappingAPI2.

A continuación se explicara el desarrollo y la lógica para desarrollar el método de unión de dos entradas.

El método se desarrolló dentro de la clase Entry de la mappingAPI2, y acepta como parámetro de entrada una Entry y devuelve una Entry.

El primer paso del método es concatenar la descripción de ambas entradas, la entrada desde la cual se llama al método y la entrada que se pasa por parámetro.

El segundo paso es quizás el más relevante del método de unión ya que se encarga de conseguir las clases que se puedan usar como vínculos entre diferentes entradas, para lograrlo se va iterando entre las externalbounds de cada una de las entradas buscando las clases que coincidan, y que no sean data types ya que al recorrer las clases clases de las external bounds, ya nos aseguramos que estas tengan externalbounds. De este paso obtenemos un listado de entradas que se usaran como unión.

Como tercer paso agregamos cada path de la segunda entry en la primera entry.

El cuarto paso de agrega las internal bounds ajustadas de la segunda entry en la nueva entrada con toda la información, para ajustar estas internalbounds, se buscan las internal bounds originales de la entrada dos y en las coordenadas de las clases se deja el mismo número pero a las coordenadas del path se le suma el valor de la longitud de los paths dela entrada original.

A pesar de haber agregado todos los internalbounds en el paso anterior en el quinto paso se agregan las nuevas internalbounds que se generan de la unión entre clases de distintas entries. Y al igual que en el paso anterior, se ajustan los valores de las coordenadas.

Vale acotar que las internalbounds que se agregaron y modificaron en los pasos cuatro y cinco, se hicieron en ambas vistas, la vista physical y la vista conceptual.

Como sexto paso se agregan las externalbounds de la entrada dos que no hayan sido usadas para unir las entradas ya que se convierten en internalbounds. Las externalbounds también se ajustan en las coordenadas de los paths y dejando con el mismo valor los valores de la coordenada de la clase.

El séptimo y último paso del método de unificar las entradas devuelve la entrada resultante, que cuenta con la información de las 2 entradas iniciales.

Un ejemplo del resultado del método anterior seria:

Si la entry1 fuese:

```
<entry>
  <description>Birthdate of Patient</description>
  <physical_view>
   <paths>
    <path>
```

```
        <class externalBound="ExternalBound1">http://obtima.org#patient</class>
        <property>http://obtima.org/vocab/patient_person_id</property>
        <class>http://obtima.org#person</class>
        <property>http://obtima.org/vocab/person_birthdate</property>
        <class externalBound="ExternalBound2">http://www.w3.org/2002/07/owl#date</class>
       </path>
     </paths>
   </physical_view>
   <conceptual_view>
     <paths>
       <path>
        <class externalBound="ExternalBound1">http://www.ifomis.org/hdot/HDOT_PM_0058</class>
        <property>http://purl.obolibrary.org/obo/BFO_0000056</property>
        <class>http://www.ifomis.org/hdot/HDOT_CORE_041</class>
        <property>http://purl.obolibrary.org/obo/BFO_0000130</property>
        <class>http://www.ifomis.org/hdot/HDOT_CORE_043</class>
        <property>has value</property>
        <class externalBound="ExternalBound2">http://www.w3.org/2001/XMLSchema#date</class>
       </path>
     </paths>
   </conceptual_view>
 </entry>
```

Y la entry2 fuese:

```
 <entry>
   <description>Gender of Patient - Male</description>
   <physical_view>
     <paths>
       <path>
        <class externalBound="ExternalBound1">http://obtima.org#patient</class>
        <property>http://obtima.org/vocab/patient_person_id</property>
        <class>http://obtima.org#person</class>
        <property>http://obtima.org/vocab/person_gender</property>
        <class externalBound="ExternalBound2"
restriction="EQUAL##MALE">http://www.w3.org/2002/07/owl#string</class>
       </path>
     </paths>
   </physical_view>
   <conceptual_view>
     <paths>
       <path>
        <class externalBound="ExternalBound1">http://www.ifomis.org/hdot/HDOT_PM_0058</class>
        <property>http://purl.obolibrary.org/obo/BFO_0000087</property>
        <class>http://purl.obolibrary.org/obo/OMRSE_00000008</class>
        <property>has stringvalue</property>
        <class externalBound="ExternalBound2">http://www.w3.org/2001/XMLSchema#string</class>
       </path>
     </paths>
   </conceptual_view>
 </entry>
```

La entrada unificada en un archivo de mapeo se vería como:

```
 <entry>
  <description>Birthdate of Patient...Gender of Patient - Male</description>
  <physical_view>
   <paths>
    <path>
     <class externalBound="ExternalBound1"  internalBound="InternalBound1">http://obtima.org#patient</class>
     <property>http://obtima.org/vocab/patient_person_id</property>
     <class>http://obtima.org#person</class>
     <property>http://obtima.org/vocab/person_birthdate</property>
     <class externalBound="ExternalBound2">http://www.w3.org/2002/07/owl#date</class>
    </path>
                    <path>
     <class internalBound="InternalBound1">http://obtima.org#patient</class>
     <property>http://obtima.org/vocab/patient_person_id</property>
     <class>http://obtima.org#person</class>
     <property>http://obtima.org/vocab/person_gender</property>
     <class externalBound="ExternalBound3"
restriction="EQUAL##MALE">http://www.w3.org/2002/07/owl#string</class>
    </path>
   </paths>
  </physical_view>
  <conceptual_view>
   <paths>
    <path>
     <class externalBound="ExternalBound1"
internalBound="InternalBound1">http://www.ifomis.org/hdot/HDOT_PM_0058</class>
     <property>http://purl.obolibrary.org/obo/BFO_0000056</property>
     <class>http://www.ifomis.org/hdot/HDOT_CORE_041</class>
     <property>http://purl.obolibrary.org/obo/BFO_0000130</property>
     <class>http://www.ifomis.org/hdot/HDOT_CORE_043</class>
     <property>has value</property>
     <class externalBound="ExternalBound2">http://www.w3.org/2001/XMLSchema#date</class>
    </path>
                    <path>
     <class internalBound="InternalBound1">http://www.ifomis.org/hdot/HDOT_PM_0058</class>
     <property>http://purl.obolibrary.org/obo/BFO_0000087</property>
     <class>http://purl.obolibrary.org/obo/OMRSE_00000008</class>
     <property>has string value</property>
     <class externalBound="ExternalBound3">http://www.w3.org/2001/XMLSchema#string</class>
    </path>
   </paths>
  </conceptual_view>
 </entry>
```

También se realizó un método muy sencillo que devuelve un booleano y recibe un string, el string que recibe es el nombre de la clase y el boolean devuelve si es un datatype. Este método se llama en la iteración del segundo paso del método de unificación de las entradas.

Por último se desarrolló un método que recibe un string con el valor de la query sparql y un listado de strings con los valores que se quieren reemplazar del select, y lo que hace es reemplazar de la query sparql el valor de cada variable del select con el valor correspondiente del listado. En caso de que algún valor sea null, el valor del select se

eliminara. Y en el otro caso de que el listado tenga menos valores que variables en el select las variables restantes de la query no se modificaran. Este último método se realizó para hacer el resultado más entendible para el usuario.

Ejemplo:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?v_0_2 ?v_0_0 ?v_1_2
WHERE{
 ?v_0_0 a <http://obtima.org#patient>.
 ?v_0_1 a <http://obtima.org#person>.
 ?v_1_0 a <http://obtima.org#patient>.
?v_1_1 a <http://obtima.org#person>.
?v_0_0 <http://obtima.org/vocab/patient_person_id>  ?v_0_1.
?v_0_1 <http://obtima.org/vocab/person_birthdate>  ?v_0_2.
?v_1_0 <http://obtima.org/vocab/patient_person_id>  ?v_1_1.
?v_1_1 <http://obtima.org/vocab/person_gender>  ?v_1_2.
FILTER(?v_1_2 = "MALE"). FILTER(?v_0_0 = ?v_1_0) }
```

Se convierte en:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT date patient gender_male
WHERE{
patient a <http://obtima.org#patient>.
?v_0_1 a <http://obtima.org#person>.
?v_1_0 a <http://obtima.org#patient>.
?v_1_1 a <http://obtima.org#person>.
patient <http://obtima.org/vocab/patient_person_id>  ?v_0_1.
?v_0_1 <http://obtima.org/vocab/person_birthdate>  date.
?v_1_0 <http://obtima.org/vocab/patient_person_id>  ?v_1_1.
?v_1_1 <http://obtima.org/vocab/person_gender>  gender_male.
FILTER(gender_male = "MALE").
FILTER(patient = ?v_1_0)
}
```

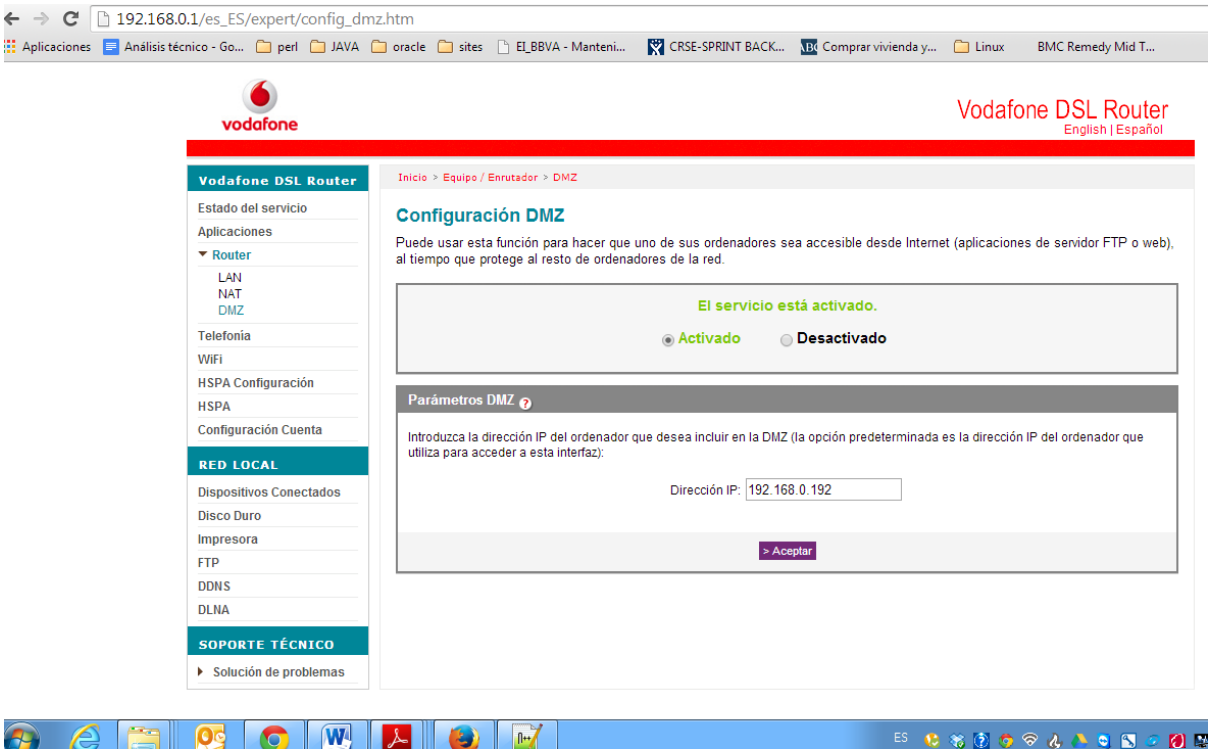
Luego de desarrollar los nuevos métodos a la mappingAPI2, se procedió al desarrollo de la interfaz web para la generación de consultas.

Se empezó a desarrollar un sistema web basado en Struts, Modelo Vista Controlador para Java, para que fuese lo más compatible con las librerías Java y cumpliera con los estándares de la industria hoy en día. Pero debido a la magnitud del proyecto p-medicine

me pareció una mejor y ams elegante solución ofrecer la mappingAPI2 a través de un servicio RESTFul para no hacer la web dependiente de una tecnología en específico, y al ofrecer el servicio en la nube, este podría ser consumido no únicamente por una web si no por cualquier otro sistema u servicio.

Para el desarrollo del servicio RESTful en un principio se realizó el desarrollo en un ordenador personal, y modificando la configuración del Router, le di acceso público al servicio.

A continuación un pantallazo dela configuración realizada en el Router.

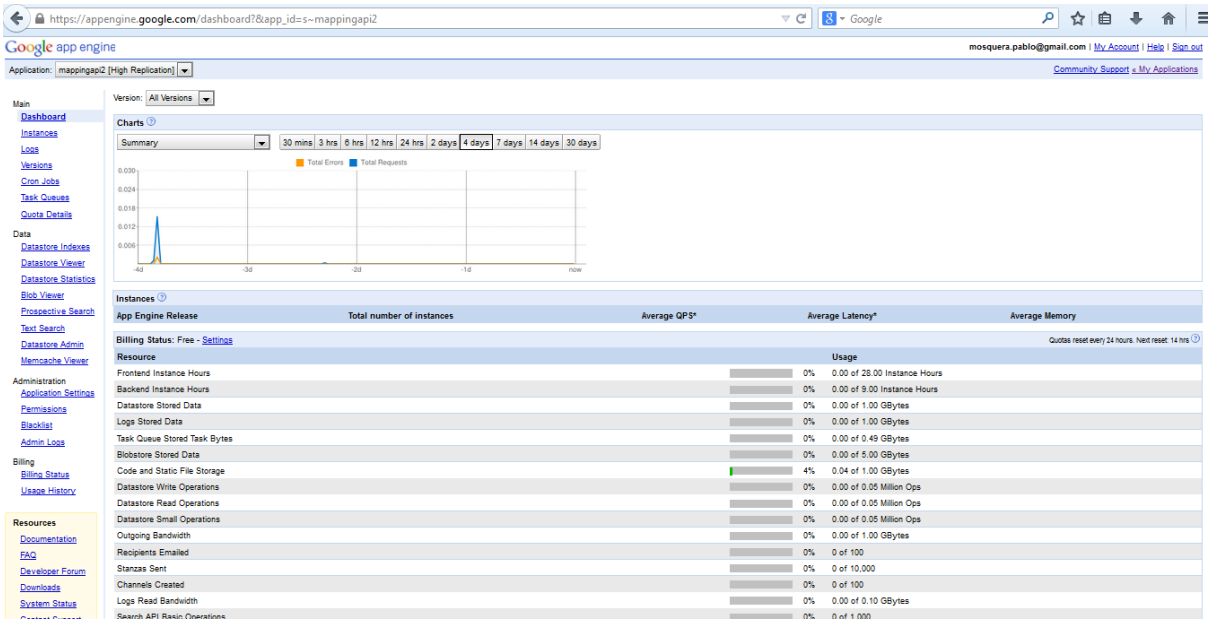


Especificamos la IP del ordenador personal al cual queríamos hacer público en la web.

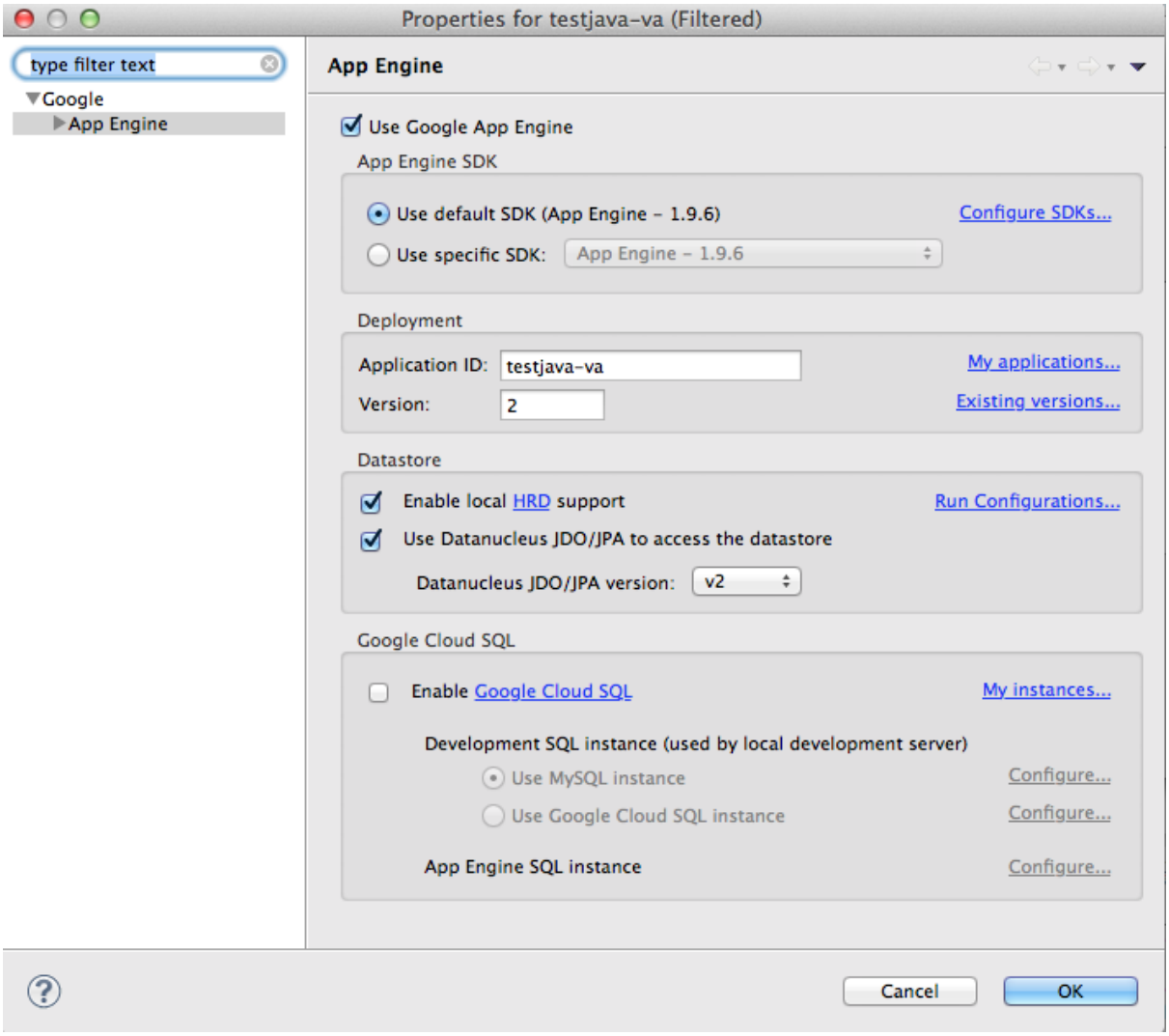
Se realizó el desarrollo en el IDE Netbeans, usando un servidor Glassfish y cuyas librerías por defecto para ofrecer un servicio REST son las librerías Jersey.

Funcionaba muy bien en local, pero decidi ofrecer el servicio desde la nube, para evitar tener el ordenador encendido para que que el servicio funcionara correctamente, por lo cual se procedi a analizar distintas Paas, para ver en cual podía ofrecer un servicio similar y que fuera gratuito.

Como opción más destacada en internet conseguí Google app Engine, conseguir realizar un servicio rest sencillo de prueba sobre la plataforma de Google. Ya que ofrecía un dashboard muy completo para la gestión de la plataforma.

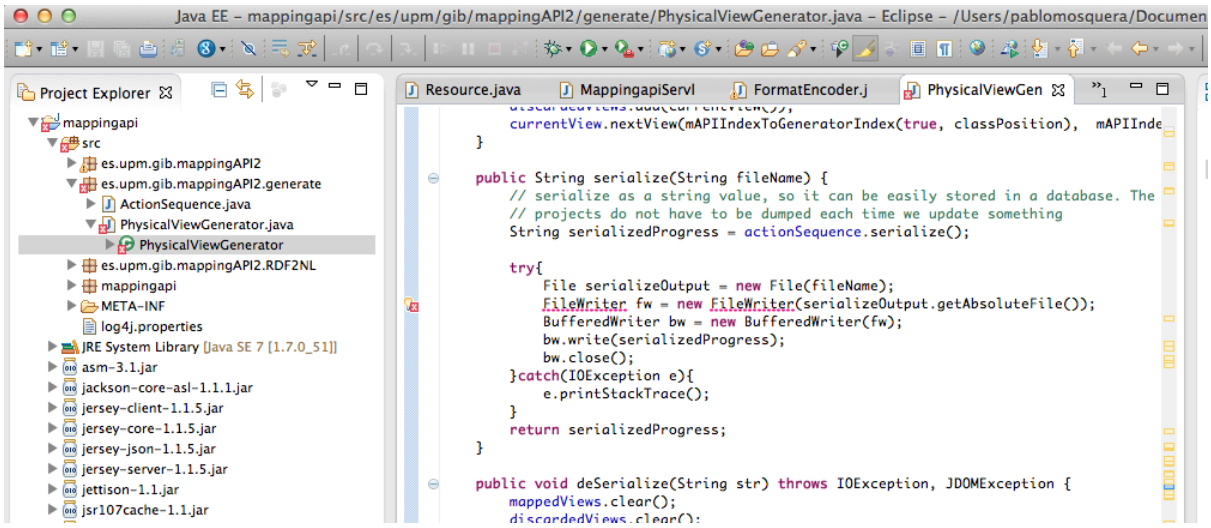


Para realizar el desarrollo y despliegue en Google App Engine, tuve que hacer uso del IDE eclipse e instalar el plugin de Google para hacer uso de la plataforma.

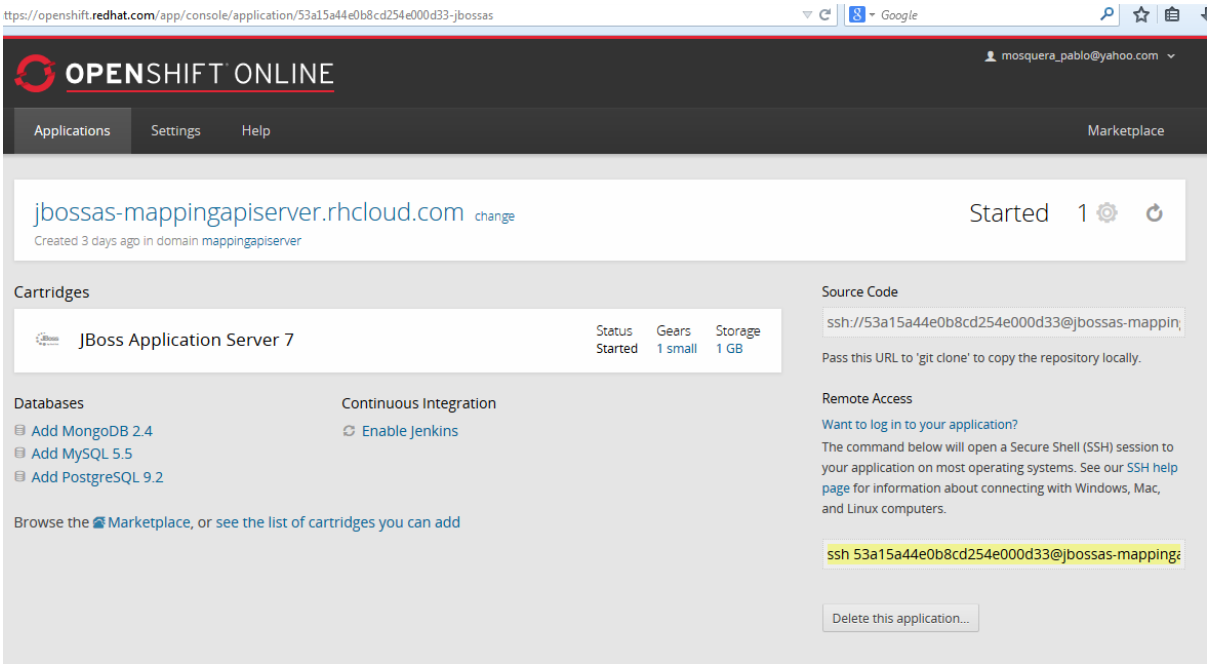


Cuando hago el desarrollo necesario para utilizar las librerías para generar el SPARQL a partir de los archivos mapping, el servicio de google app Engine (GAE), no servicio ya que la plataforma no permite lectura y escritura de archivos, se tenía que hacer uso de otros servicios de google para leer y escribir archivos, al toparme con este problema decidi buscar otra plataforma PaaS que me permitiera leer y escribir archivos, ya se hubiese tenido que hacer muchos cambios en la API para hacerla funcional a los servicios de google. Haciéndola dependiente de una plataforma en específico, lo cual a mi punto de vista no era lo más correcto.

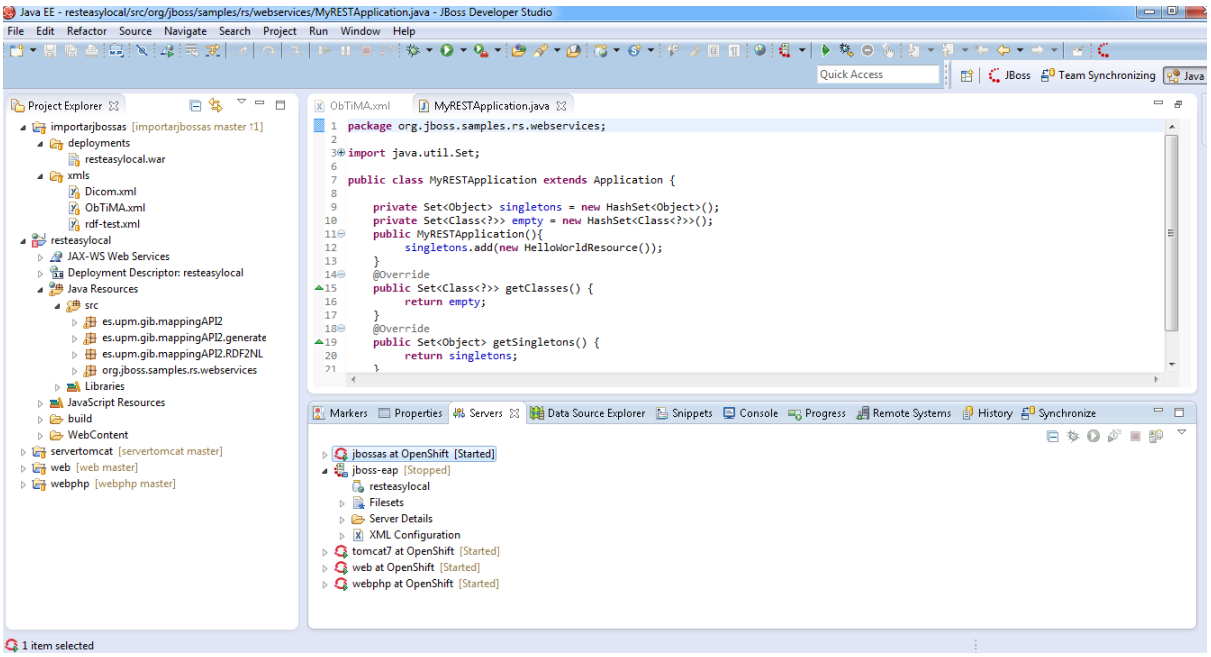
A continuación un pantallazo del error que me devolvía



Luego decidí subirla a otra plataforma PaaS, esta vez la plataforma por la que opte fue la de openshift, de Red Hat. La plataforma ofrece variedad de soluciones. La mejor valorada por los usuarios es la que hacía uso de un servidor JBoss, asi que fue por la cual opte.

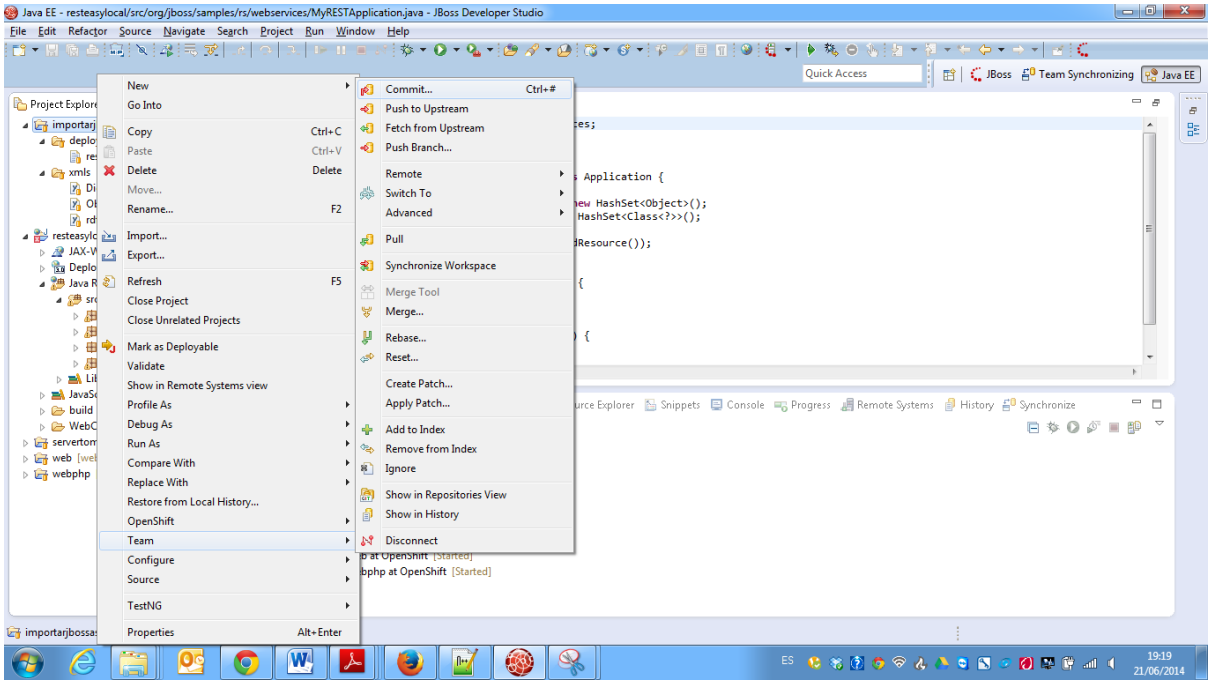


En un primer intento de despliegue del servicio, me encontré con un problema, y es que el servidor JBoss trabaja por defecto con la librería RestEasy y no con jersey la cual había estado utilizando, por lo cual fue necesario hacer un cambio de librerías y de IDE para poder trabajar cómodamente.

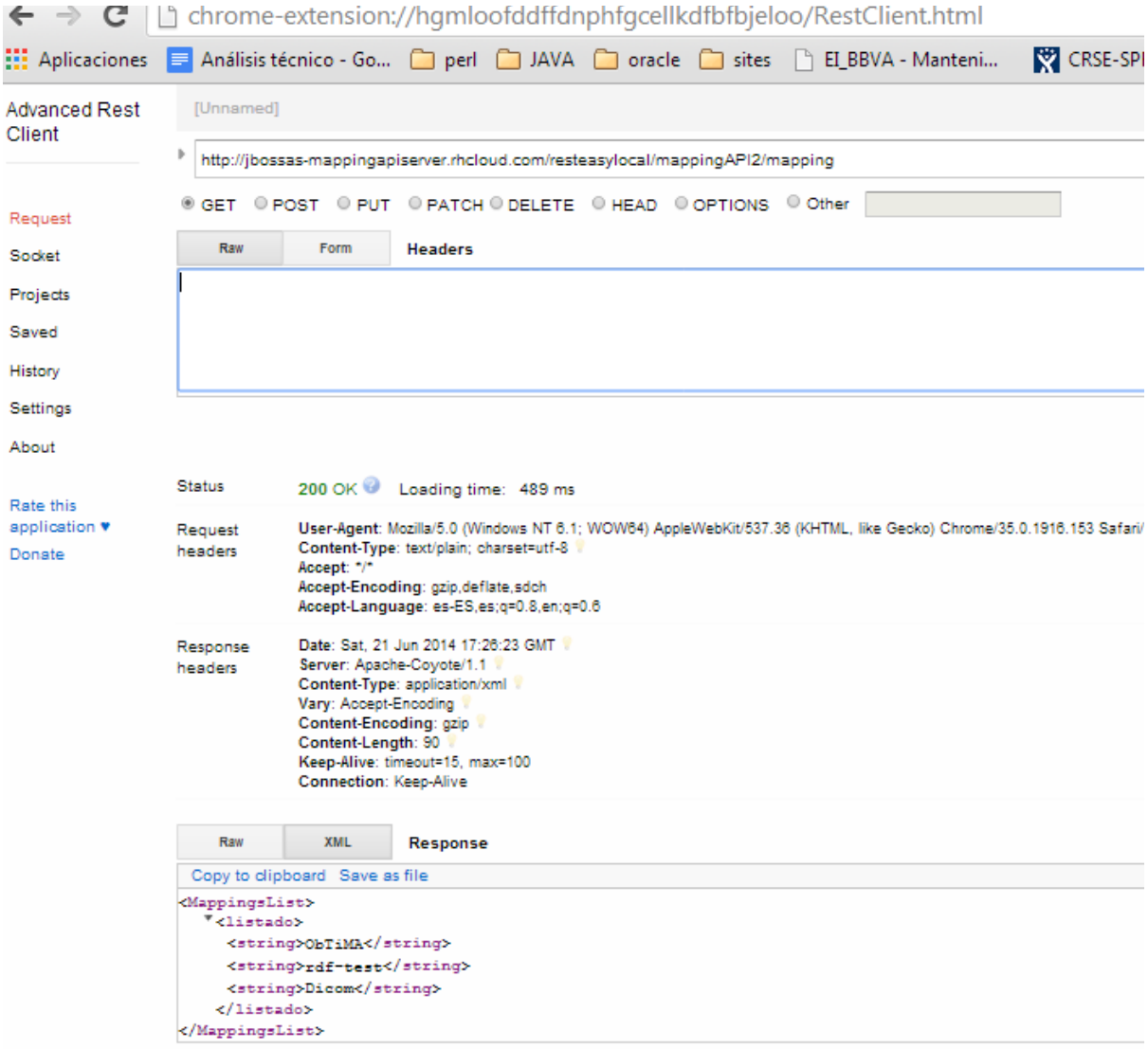


Una vez realizado el cambio a la librería RestEasy y el haber cambiado de IDE de Netbeans a un eclipse de Red Hat con las opciones para desplegar en openshift ya configuradas, el despliegue del servicio con las funcionalidades de la mappingAPI2 correspondientes fue el correcto y funcionaba perfectamente.

El despliegue de la aplicación en openshift se hace a través de GIT, que a su vez también sirve como repositorio.

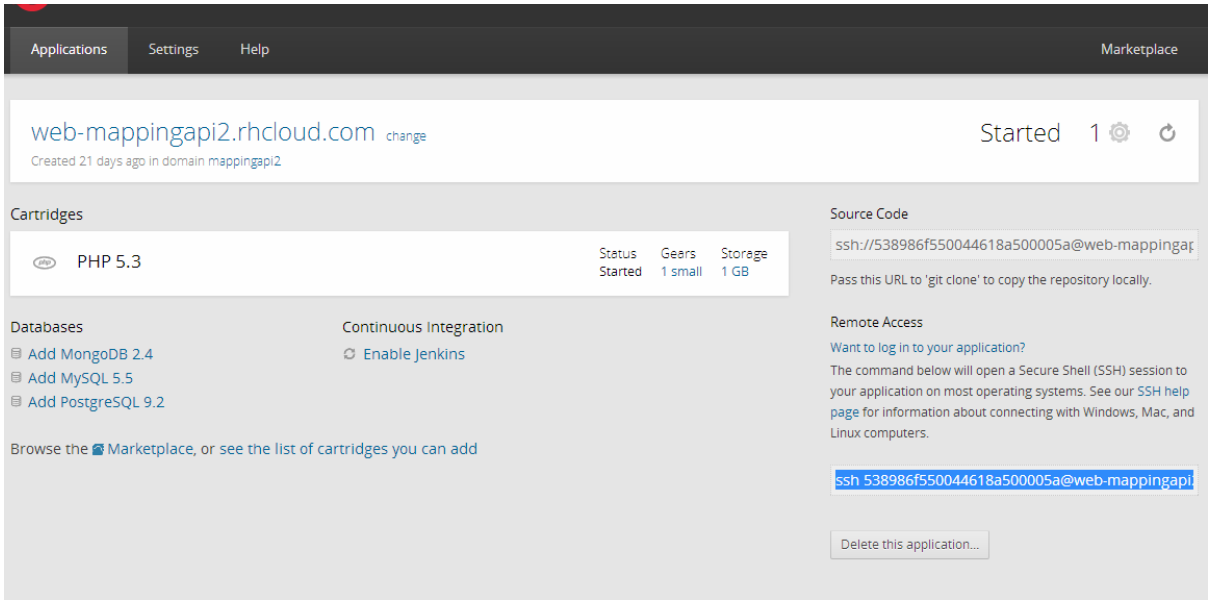


Realice pruebas para verificar el correcto funcionamiento del servicio. A continuación un pantallazo de una de las pruebas del servicio.



Una vez probado el servicio se procedió a realizar la Interfaz de Usuario Web para poder generar consultas a través de una página web.

Por lo cual, se decidido usar la misma plataforma como servicio, openshift, porque ofrece una variedad de plataformas web ya configuradas para trabajar y ya se contaba con la experiencia anterior de haber montado el servicio RESTful.

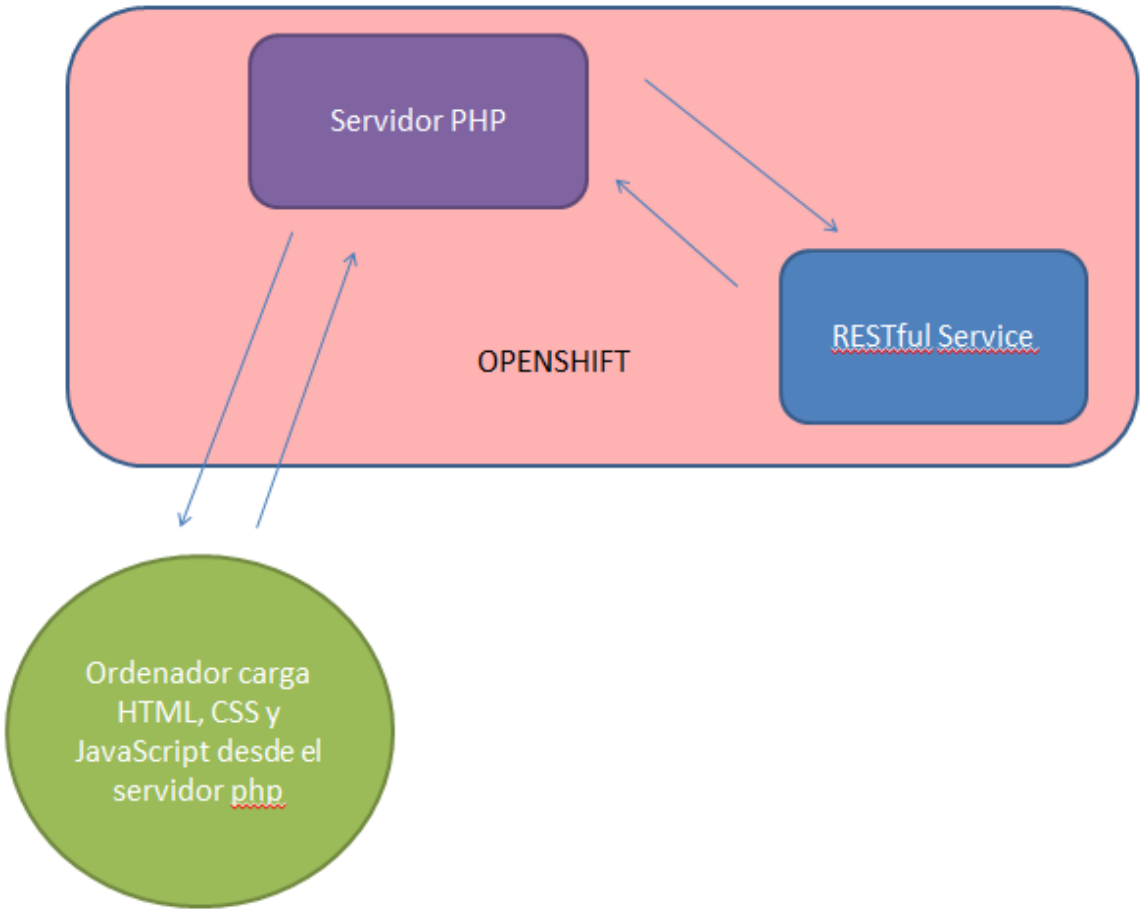


Se decidió hacer uso de una plataforma que tuviese las siguientes tecnologías PHP, HTML, JavaScript y CSS. Ya que son un conjunto de herramientas web muy usadas y más que probadas que funcionan muy bien interactuando juntas.

Se hace uso de HTML y CSS para darle forma a la página web, ya que html define cada parte del archivo y CSS le da formato a la página web en sí.

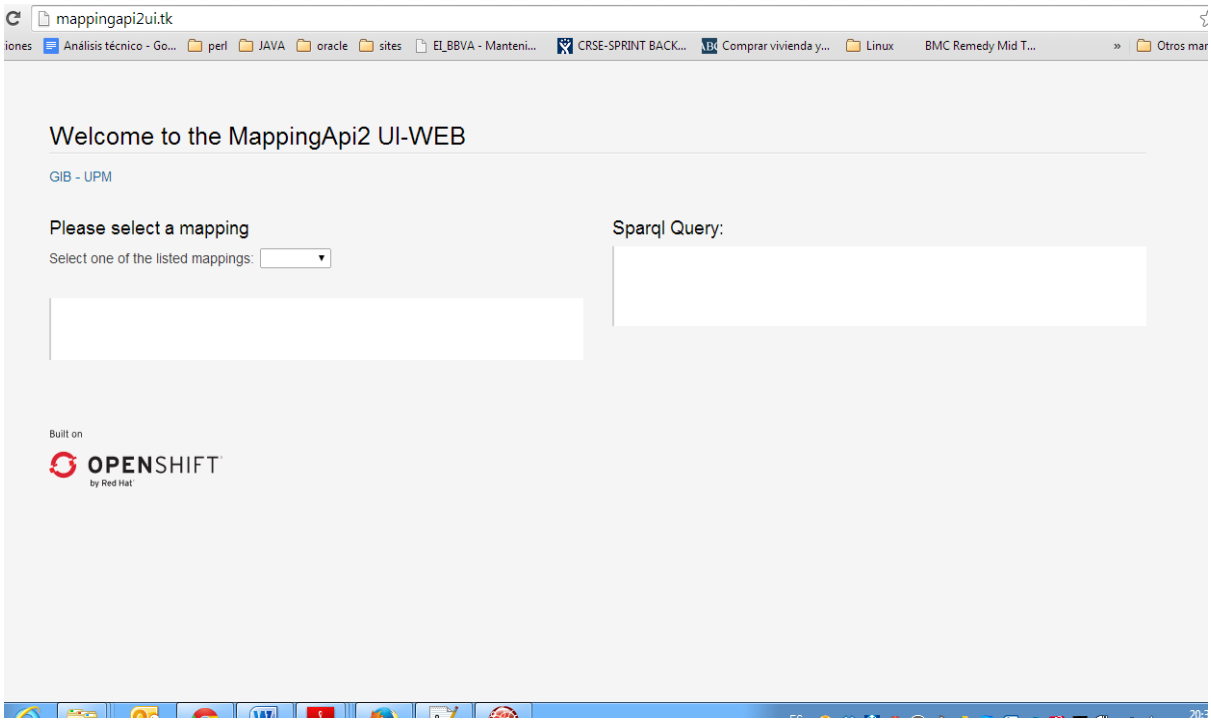
Javascript para hacer dinámica la página web y realizar las llamadas al archivo PHP que este a su vez se encarga se hacer las llamadas al servicio REST y parsear alguna información para luego enviar la respuesta al javascript, este último paso se realizó porque a veces hay ordenadores que no funcionan bien con llamadas de javascript directamente al servidor haciendo más lento el procesamiento y cargado de la página, con PHP hacemos que parte del procesamiento lo haga el servidor.

A continuación el diagrama representa como se realiza la comunicación:

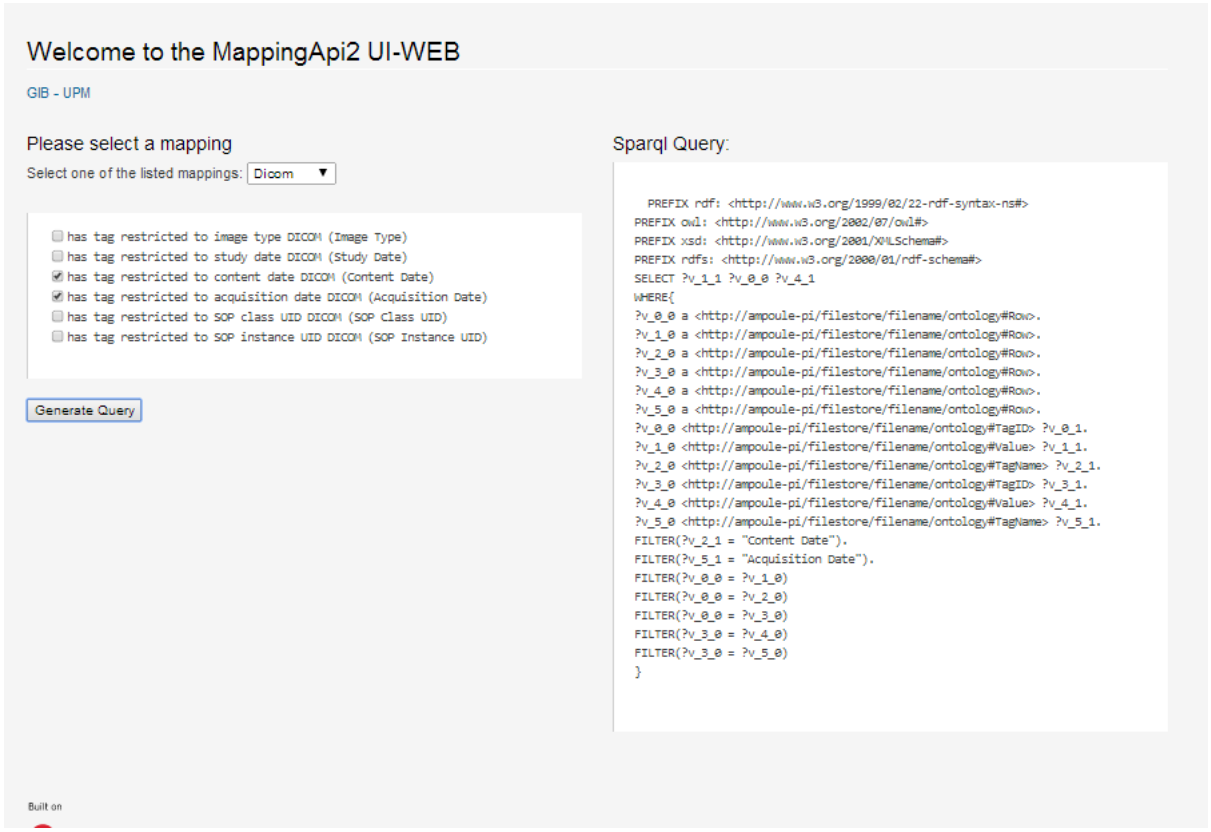


El servidor PHP y el servicio RESTful están en la PaaS de OpenShift. El ordenador se descarga el HTML, CSS y Javascript del servidor PHP y los métodos en Javascript se encargan de comunicarse con los métodos php y estos a su vez se comunicación el servidor que contiene el servicio REST.

La página web se carga una sola vez.



Y luego haciendo uso de Javascript y librerías como Jquery hacemos el intercambio de información y actualizamos la información en la página.



Se dio de alta un dominio gratuito para hacer la dirección URL de la página web más sencilla de recordar, La URL es: http://www.mappingapi2ui.tk/

Al tener la MappingAPI2 soportada en un servicio RESTFul, decidi realizar una aplicacion móvil que tuviese las funcionalidad de la página web. Para la realización de las aplicaciones móviles se usó el framework de desarrollo Phonegap, permitiendo obtener aplicaciones para distintas plataformas con un único desarrollo.

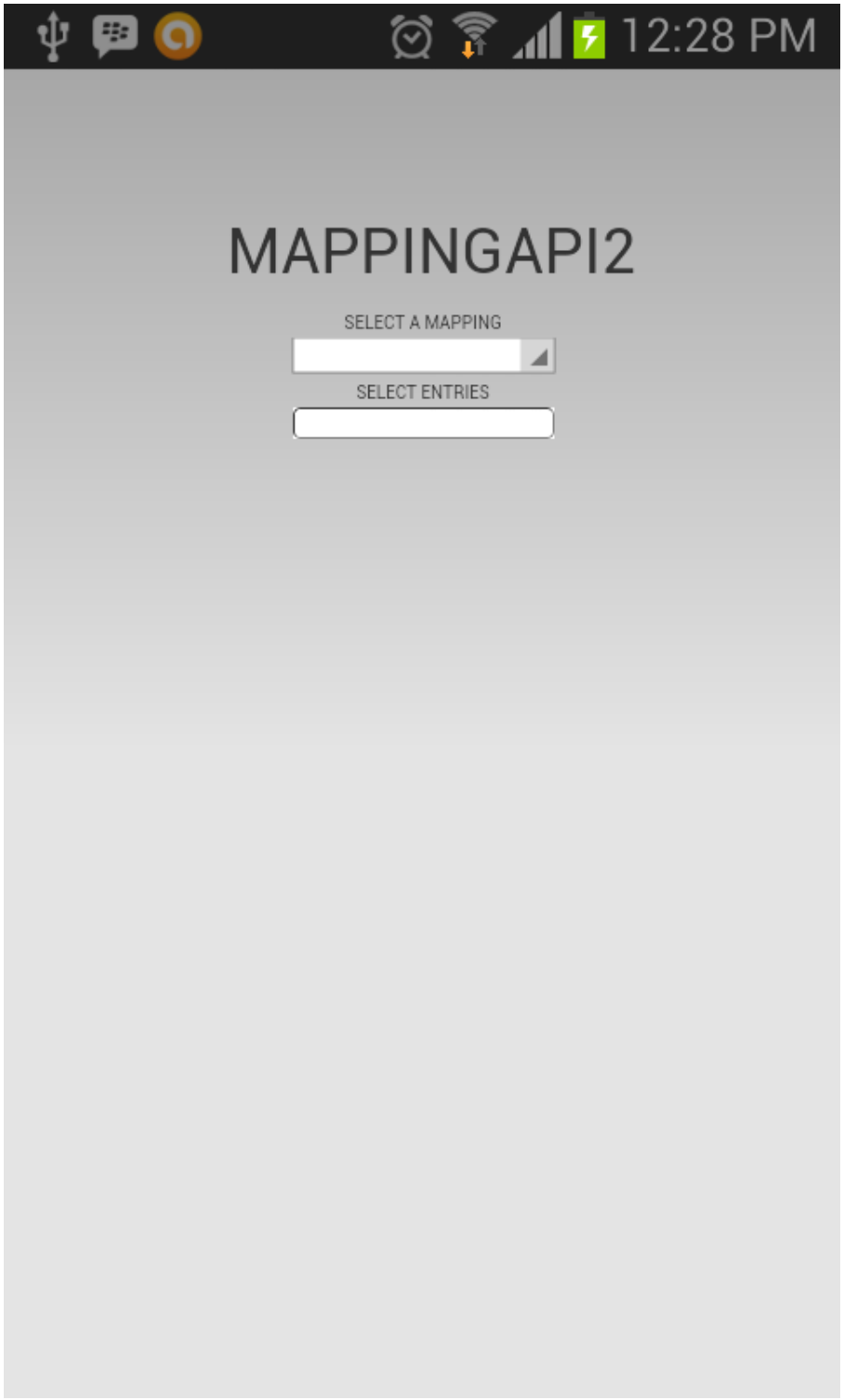
El desarrollo se llevó a cabo utilizando XCode para IOS, la plataforma permite realizar aplicaciones para productos Apple, a su vez es compatible con el framework phonegap para aplicaciones móviles. XCode junto con su emulador, me permitió realizar el desarrollo de manera eficaz a la vez que hacia las pruebas.

Cuando se tuvo la versión de la aplicación móvil funcionando correctamente, se procedió a usar el compilador que tiene Adobe online, que con subir el proyecto realizado en alguna plataforma te genera el resto.

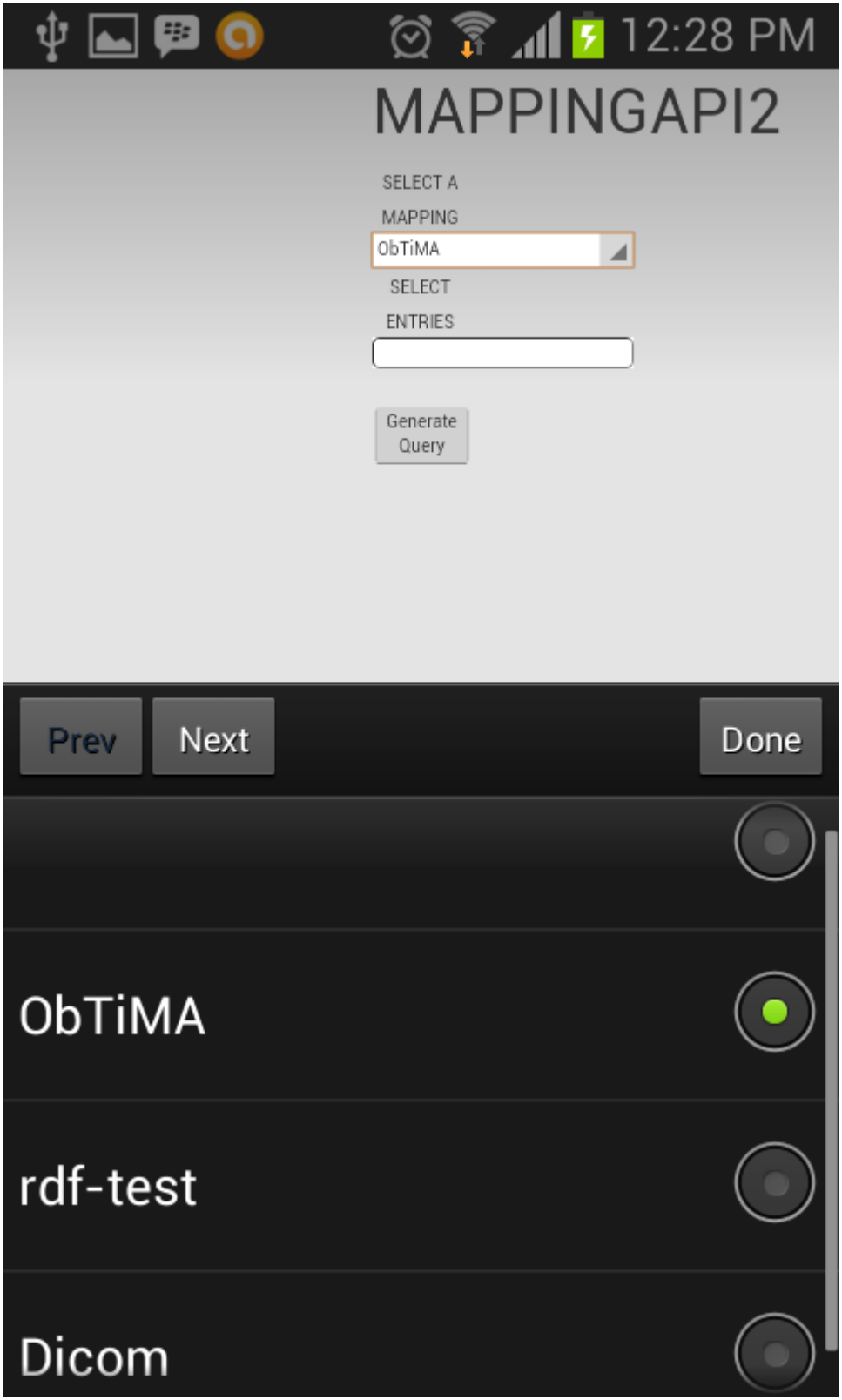
Se tienen 3 versiones de la app móvil, para IOS, Android y Windows Phone.

Android:

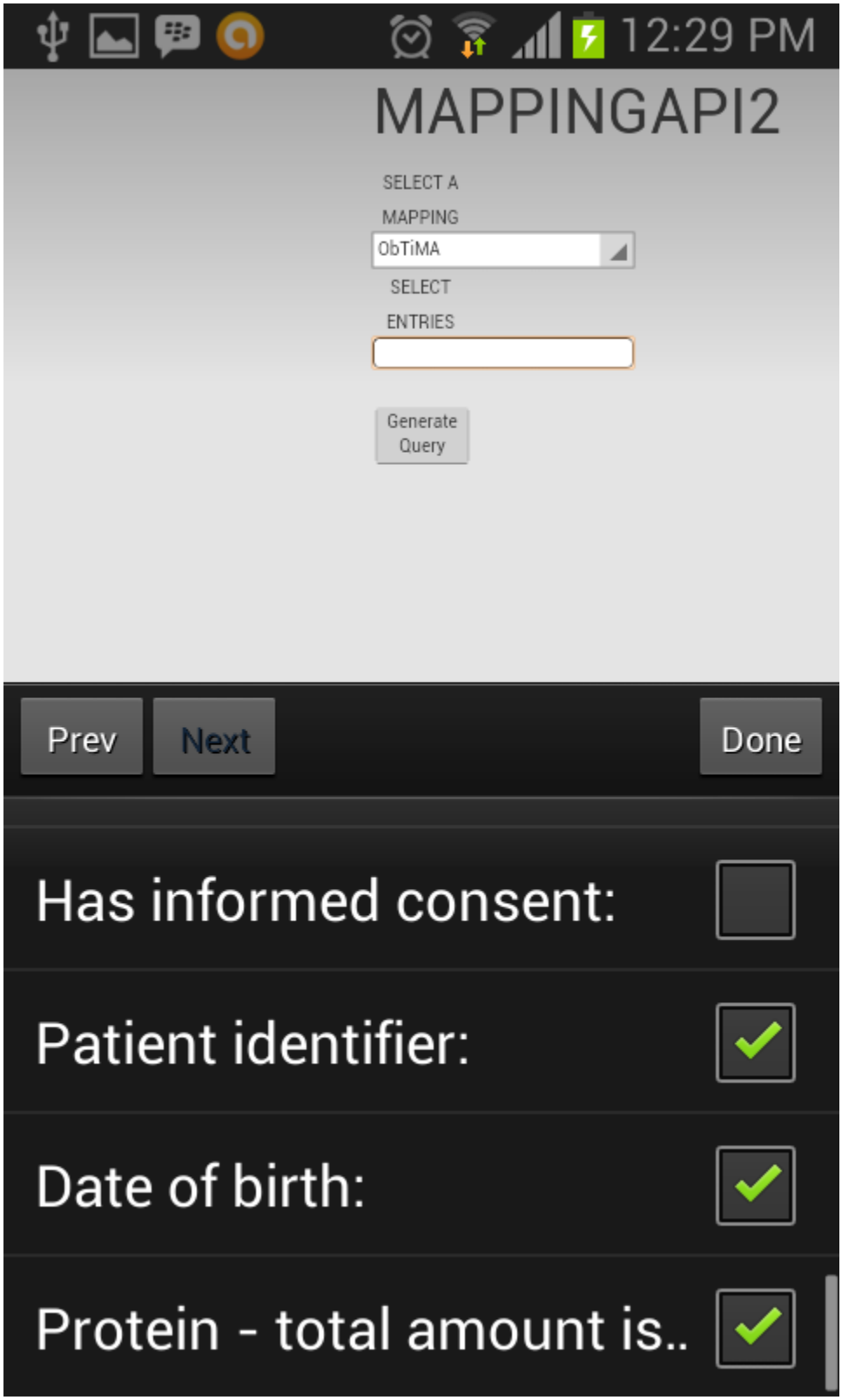
Modo de Uso.



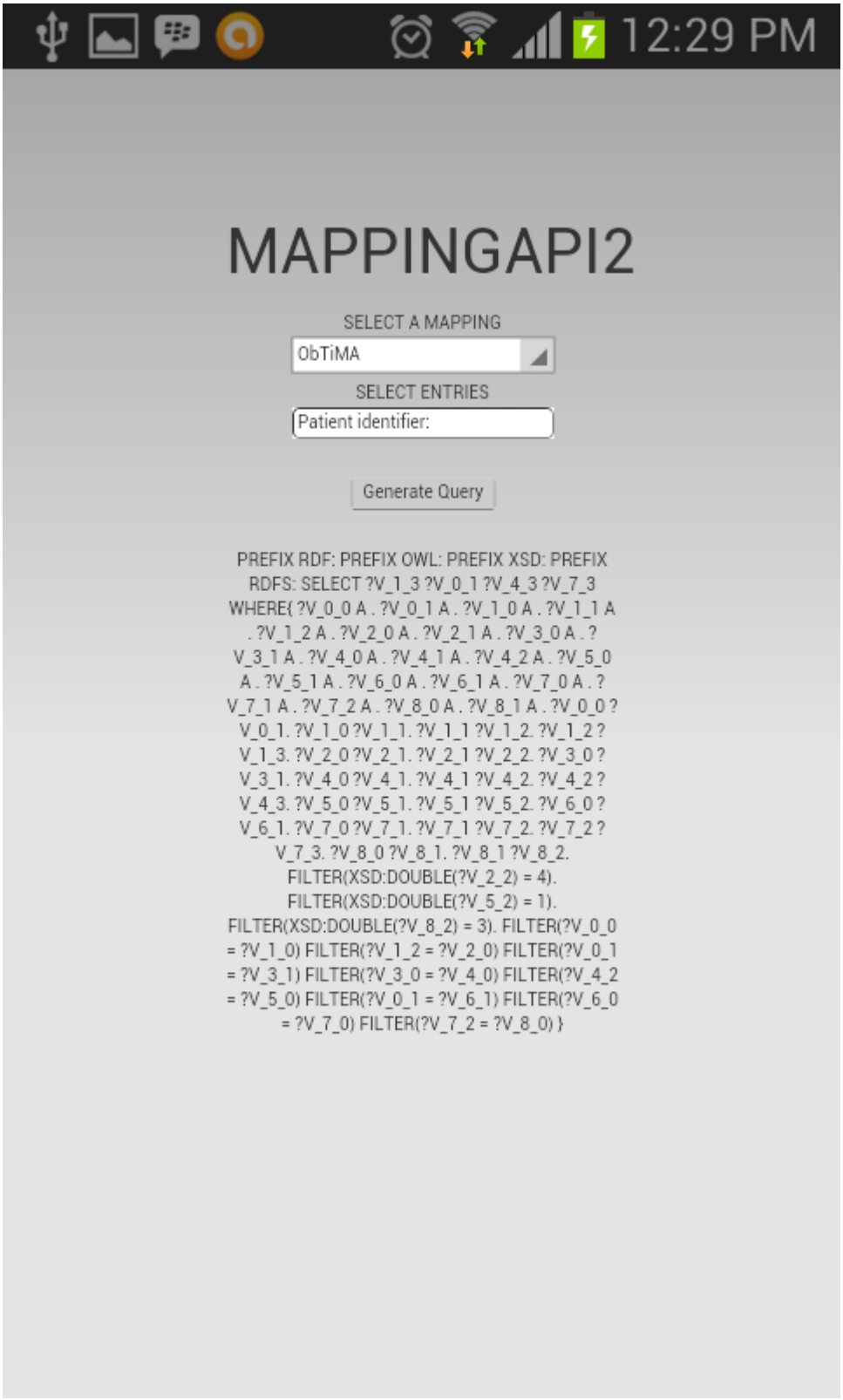
Se selecciona un Archivo de Mapeo, actualmente solo deja seleccionar un archivo de mapeo para cada consulta.



Una vez seleccionado el mapeo se procede a seleccionar las entradas que se quieran consultar, se pueden seleccionar multiples entradas.



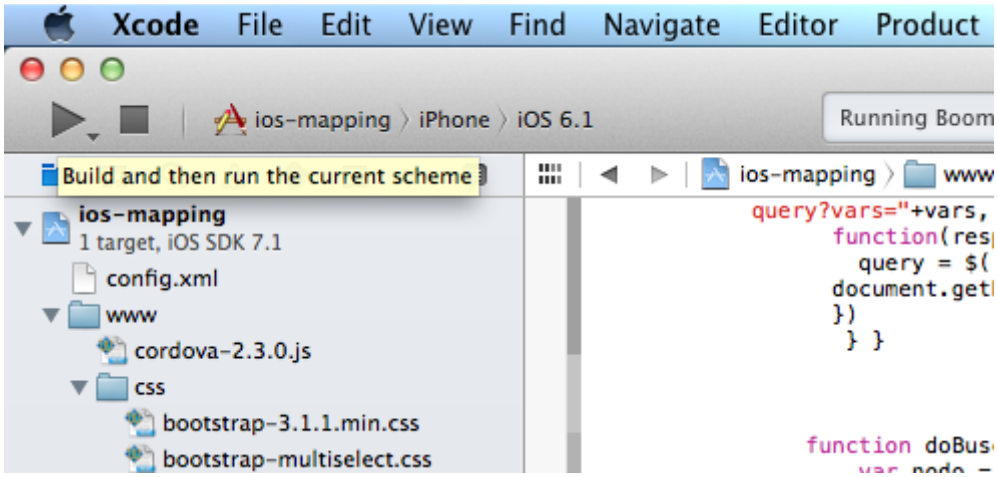
Cuando ya hayan sido seleccionadas, se pincha en el botón Generate Query para que la aplicación móvil se conecte al servicio RESTFul y el servicio le devuelva el query SPARQL generado.



La aplicación para IOS no se pudo probar directamente en un dispositivo móvil, debido a que para compilar se debe dar de alta al usuario en el programa de desarrollo de Apple, darse de alta cuesta

aproximadamente 100 $, o desbloquear un dispositivo móvil de Apple. Ninguna de las dos opciones fueron posibles para el proyecto, pero gracias al Emulador se pudo probar el funcionamiento de la aplicación.

Primero se pincha en el botón de play, seleccionando el dispositivo Apple que se quiera emular,



Se carga la aplicación y seleccionamos un archivo de mapeo, actualmente solo se permite la selección de un archivo de mapeo.

Una vez seleccionado el archivo se autocompleta la sección de entradas, permitiendo seleccionar todas las entradas que se quieran para la generación de la query.

Una vez seleccionadas las entradas, se hace click en el botón Generate Query para que se genere la consulta SPARQL asociada a las entradas del mapping, cabe acotar que la lógica no es realizada desde el móvil, ya que la aplicación se conecta al servicio RESTFul de la mappingAPI2.


Carrier
5:54 AM
MAPPINGAPI2
SELECT A MAPPING
ObTiMA
SELECT ENTRIES
3 Items
Generate Query
PREFIX RDF: PREFIX OWL: PREFIX XSD: PREFIX RDFS: SELECT ?V_0_2 ?V_0_0 ?V_1_2 ?V_2_2 WHERE{ ?V_0_0 A . ?V_0_1 A . ?V_1_0 A . ?V_1_1 A . ?V_2_0 A . ?V_2_1 A . ?V_0_0 ?V_0_1. ?V_0_1 ?V_0_2. ?V_1_0 ?V_1_1. ?V_1_1 ?V_1_2. ?V_2_0 ?V_2_1. ?V_2_1 ?V_2_2. FILTER(?V_1_2 = "MALE"). FILTER(?V_0_0 = ?V_1_0) FILTER(?V_0_0 = ?V_2_0) }

5 Conclusiones:

En este trabajo de fin de master se ha diseñado, elaborado e implantado una solución que expande la usabilidad de la mappingAPI2 permitiendo la generación de consultas cada vez más elaboradas, complejas y haciéndola a su vez más accesible y basándose en tecnologías que se usan en el ámbito profesional para la resolución de proyectos similares. Logrando satisfacer los objetivos de Realizar un desarrollo para expandir la funcionalidad de la API MAppingAPI2, Elaboración del prototipo y la Implantación de la solución.

Para lograr la expansión de la mappingAPI2, se hizo hincapié en el aprendizaje de la API para buscar la manera más efectiva de agregar la nueva funcionalidad. Se modificó la API para obtener las funcionalidades que se deseaban sobre expandir generación de las consultas. Haciendo posible la consecución de los objetivos de Estudio de la MAppingAPI2.

Importante destacar que fue de vital importancia el estudio del lenguaje SPARQL, el cual llevo al logro del objetivo del análisis de la estructuración del lenguaje.

Parte de trabajo final era darle un interfaz a la API, preferiblemente web, para que se hiciera un uso efectivo de la misma, no solo se realizó la interfaz web que permite realizar consultas, sino que también se realizó un servicio RESTful para que su aprovechamiento fuese mucho mayor. El servicio es usado por la web para la generación de las consultas.

Los servicios RESTful son los servicios más comunes y fáciles de usar hoy en día en la web, permiten su consumo por prácticamente cualquier plataforma con acceso a la web, gracias a su uso, se consiguió el objetivo de ofrecer la información para su posterior consumo.

Se buscó la manera de mitigar los grandes riesgos antes de empezar con el desarrollo e implantación del proyecto de master. Y también se solucionaron los problemas que fueron ocurriendo a medida que se fue realizando la tesis.

Es importante que proyectos de informática biomédica se sigan realizando y apoyando debido al potencial que tienen y a la ayuda que aportaran a futuro, como se explicó en la introducción la necesidad de unificar la información es latente y al momento de lograr dicha unificación en el caso específico del proyecto p-medicine se logra no únicamente consultar información heterogénea sino que se mejora y se adapta el tipo de tratamiento a un paciente logrando que el tratamiento sea a medida para lograr mejores resultados.

6 Líneas Futuras

A futuro de aconseja ampliar en la web la opción de poder elegir entradas de diferentes ficheros de mapeos, para poder conseguir opciones más interesantes de búsqueda, esta ampliación de la opción no sería difícil de implementar, ya que la modificación que se realizó a la mappingAPI2 acepta entradas de distintos ficheros de mapeos. Por razones de tiempo y del alcance propuesto para del proyecto no se realizó en este trabajo de fin de Master.

Sería de gran interés implementar una opción en la web que permita realizar la consulta de la query SPARQL generada en el sistema p-medicine, evitando que el usuario tenga que copiar la consulta y exportarla al sistema. Resolviendo de esta manera un futuro riesgo operacional por parte de los usuario, por riesgo operacional hacemos referencia al riego que existe en caso de que el usuario pueda equivocarse al momento de exportar el query al sistema de consulta que tiene p-medicine.

Por último es aconsejable que al tener un servicio que puede ser consumido por diferentes plataformas, el mismo se siga mejorando, para que de esta manera el resto de plataformas mejoren también haciendo un único esfuerzo en la lógica, ya que esta se encuentra centralizada, donde los únicos cambios serán en la apariencia de las aplicaciones para que puedan hacer uso de las nuevas mejoras.

7 Bibliografía:

[1] Web P-MEDICINE – p-medicine.org Recuperado 14 de junio de 2014.

[2] Web GIB fi UPM – Grupo de Informatica Biomedica gib.fi.upm.es Recuperado 10de junio de 2014

[3] Análisis del desarrollo de aplicaciones móviles multi plataforma – e-archivo.uc3m.es /bitstream/10016/15514/1/PFC_Diego_Pinedo_Escribado_(versión_reducida_Espanol).p df – Recuperado el 15 de mayo de 2014

[4] IDE - http://www.buenastareas.com/ensayos/Que-Es-Un-Ide-De-Programacion/163537.html - Recuperado 17 de mayo de 2014

[5] w3c - http://www.w3.org/TR/wsdl - Recuperado 05 de mayo de 2014

[6] w3c - http://www.w3.org/TR/soap12-part1/ - Recuperado 05 de mayo de 2014

[7] w3c - http://www.w3.org/RDF/ - Recuperado 05 de mayo de 2014

[8] w3c - http://www.w3.org/html/ - Recuperado 05 de mayo de 2014

[9] w3c - http://www.w3.org/XML/ - Recuperado 05 de mayo de 2014

[10] GIT - git-scm.com - Recuperado 05 de mayo de 2014

[11] Alegsa - alegsa.com - Recuperado 05 de mayo de 2014

[12] XML – w3schools.com/XML - Recuperado 05 de mayo de 2014

[13] CSS – maestrosdelweb.com - Recuperado 05 de mayo de 2014

[14] JAVA – java.com - Recuperado 05 de mayo de 2014

[15] J2EE – webopedia.com/TERM/J/J2EE - Recuperado 05 de mayo de 2014

[16] WAR – myjavazone.com/2012/07/archivos-war.html - Recuperado 05 de mayo de 2014

[17] RestFul – w3.org/TR/web_arch - Recuperado 05 de mayo de 2014